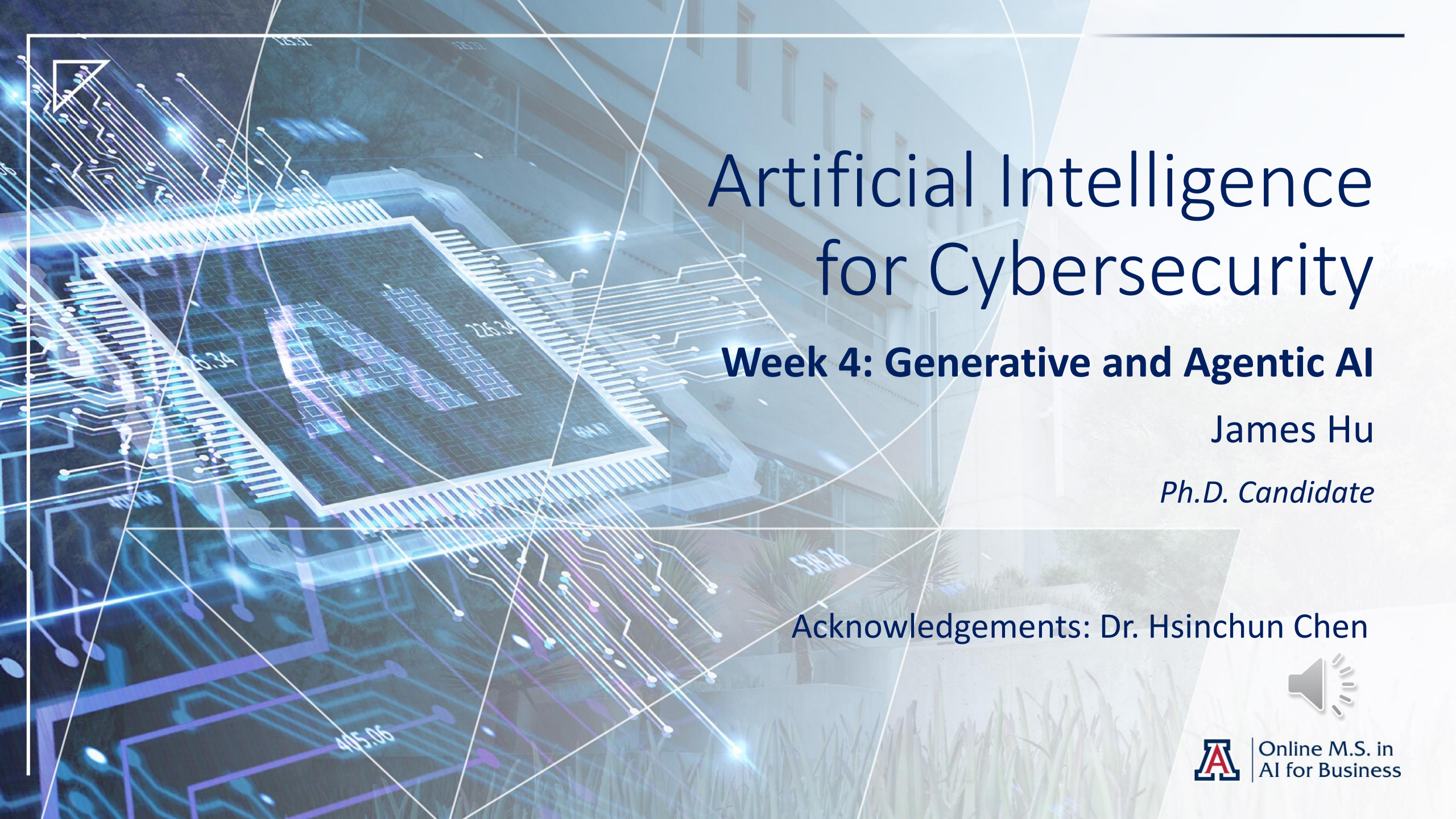




Artificial Intelligence for Cybersecurity

Week 4: Generative and Agentic AI

James Hu

Ph.D. Candidate

Acknowledgements: Dr. Hsinchun Chen



Online M.S. in
AI for Business



Today's Learning Objectives

- ▶ Understand the fundamental difference between LLM-based and Agentic AI (AAI)-based systems.
- ▶ Explain the three core capabilities of AAI agents.
- ▶ Explain the difference between static and dynamic orchestration in multi-agent AAI systems
- ▶ Understand how OpenClaw operationalizes the AAI techniques covered with a focus on continual operation.



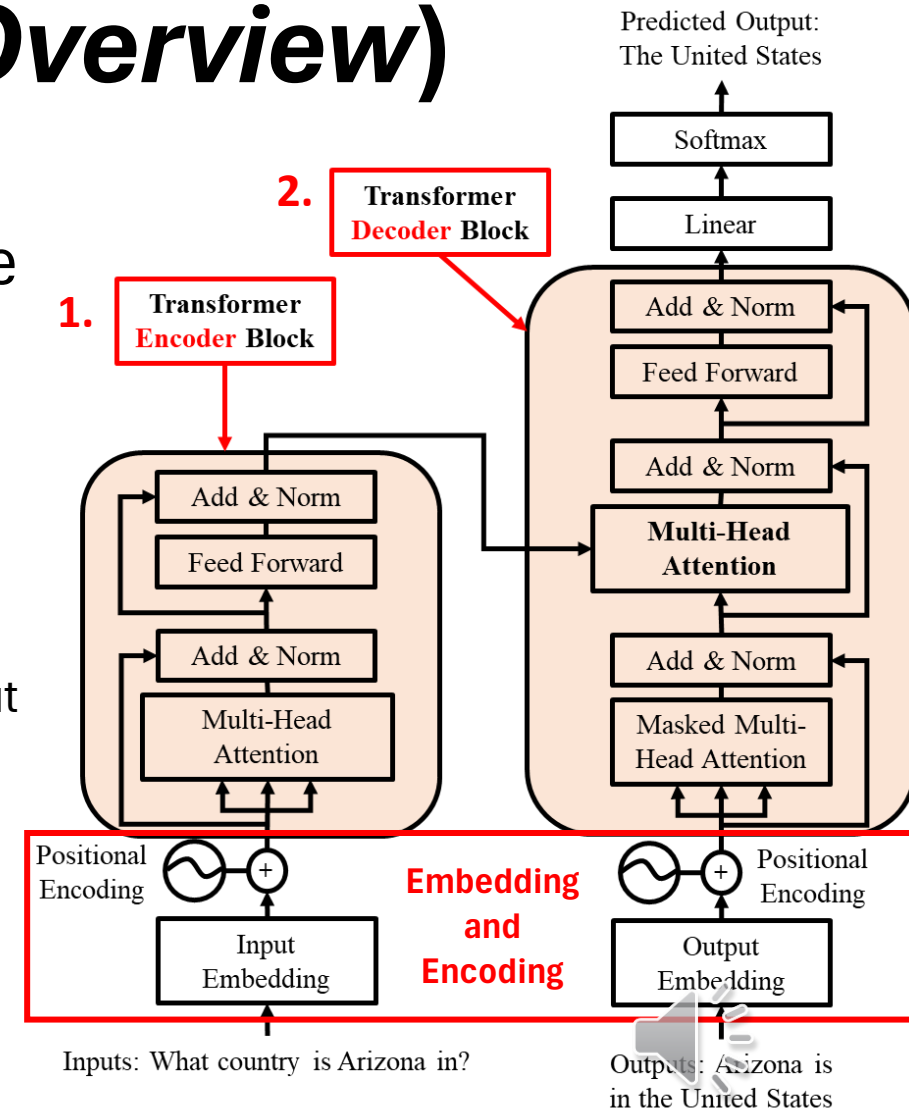
From Week 2: Key to ML Effectiveness

- ▶ Fundamentally, all AI models are math equations that extract a dataset's patterns into its parameters.
- ▶ Digitalization has increased the volume and complexity of data.
 - ▶ Larger and more complex datasets contain more comprehensive and complicated patterns.
- ▶ To be able to learn these patterns, AI models must also become more complicated.



From Week 3: Transformer (Overview)

- ▶ **Transformer** models are designed to process entire text snippets simultaneously.
 - ▶ **Input (Embedding and Encoding)**: Textual datapoints tokenized with positional encodings
 - ▶ **Output**: Two major components:
 1. **Encoder Block**: Create latent representation of text
 2. **Decoder Block**: Decode latent representation for task output
 - ▶ **Loss Function**: Typically, the cross-entropy loss
 - ▶ **Parameters, and Optimization** : Same as ANNs.
- ▶ Increased complexity from scaled Transformer architecture allows Large Language Models (LLMs) to “understand” natural language.



Current Landscape of LLMs

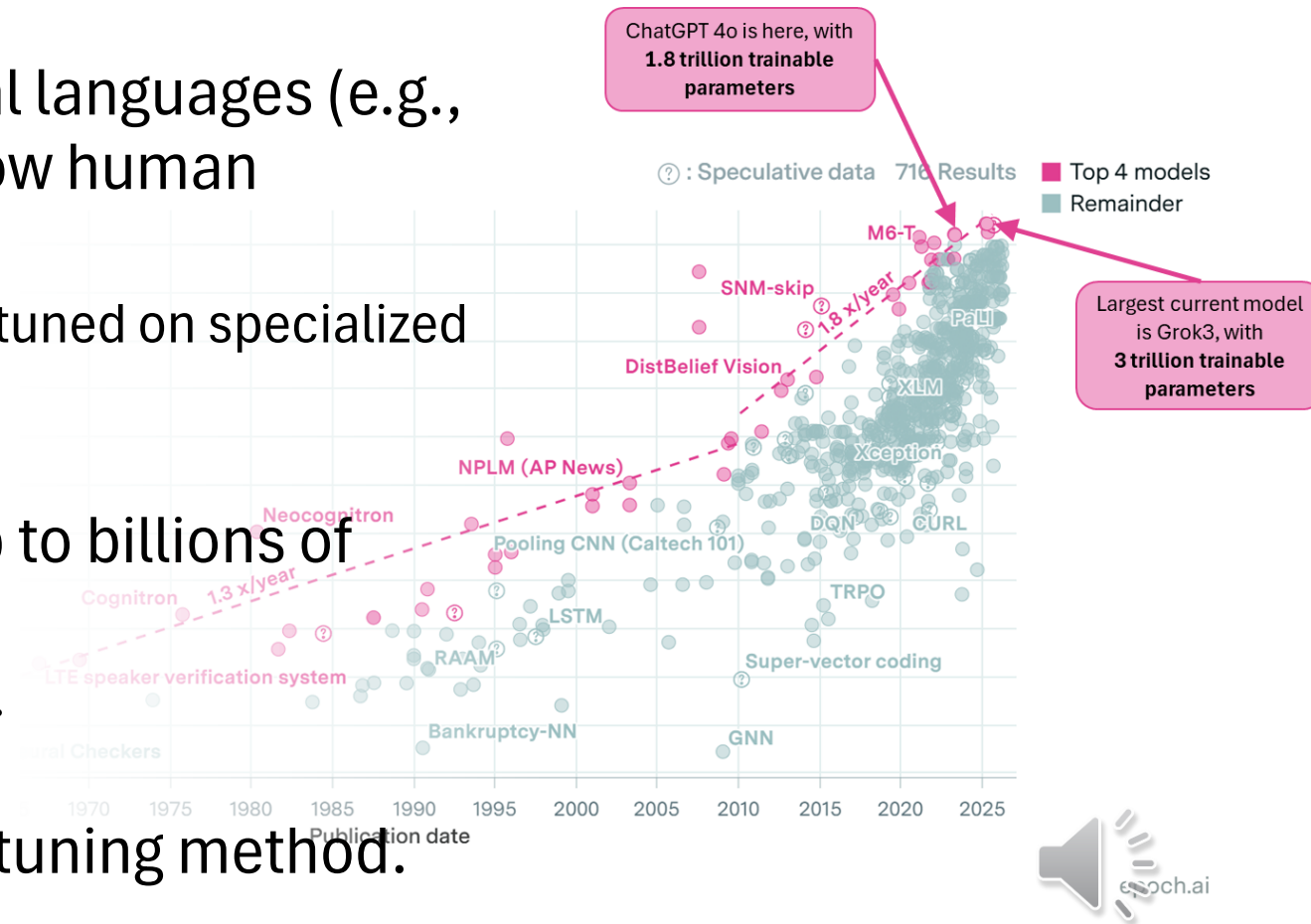
- Access to pre-trained LLMs typically fall under two types:

- Closed-Source:** Only allow users to interact with LLM via web applications or APIs
- Open-Source:** Allow users to download full model weights and architectures

Access	Company	Flagship Model(s)	Parameters (Billions)	Best Capabilities	GPU Load to Run (GB)
Closed-Source	OpenAI	GPT-5.5 	2,500	Agents, General Chat	1,400
	Alphabet (Google)	Gemini 2.5 Pro 	3,000	Agents, Multi-Modal	6,000
	Anthropic	Claude Opus 4.7 	800	Agents, Coding	1,600
	xAI	Grok 5 	1,500	Social Media	3,000
Open-Source	Meta	Llama4 Scout 	109	General Chat	220
	Mistral AI	Mistral Medium 3.5 	1,000	Efficient Running	2,000
	Moonshot AI	Kimi K2 	1,000	Database Analysis	2,000
	DeepSeek AI	DeepSeek-R1 	671	Efficient Coding	1,350
	Alibaba	Qwen 3.6 	1,000	Math, Image Processing	2,000

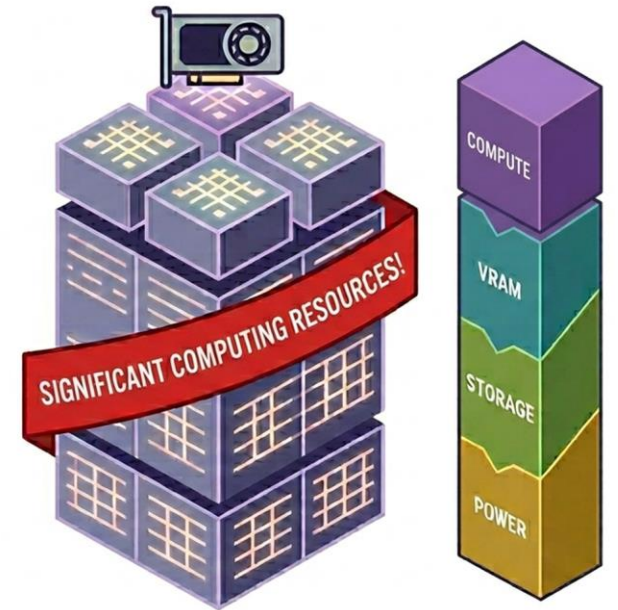
From Week 3: Pretrained and Task Adaption

- ▶ LLMs are first pretrained with natural languages (e.g., Reddit, Wikipedia) to understand how human languages work.
 - ▶ These pretrained models are then finetuned on specialized data for certain tasks.
- ▶ However, modern LLMs can have up to billions of parameters.
 - ▶ It's not realistic to finetune all of them.
- ▶ We need a parameter-efficient fine-tuning method.
 - ▶ But is that even enough???



Current Landscape of LLMs

- ▶ Even running modern LLMs require significant computing resources.
 - ▶ An Nvidia RTX 5090 only provides 32GB of GPU load.
- ▶ Finetuning an LLM can require up to a trillion times more GPU load than running it!!
 - ▶ Also requires access to an LLM's parameter, but the most capable LLMs are often closed-source.
- ▶ While techniques like quantization help reduce required GPU loads, training LLMs remains an expensive option for task adaption.

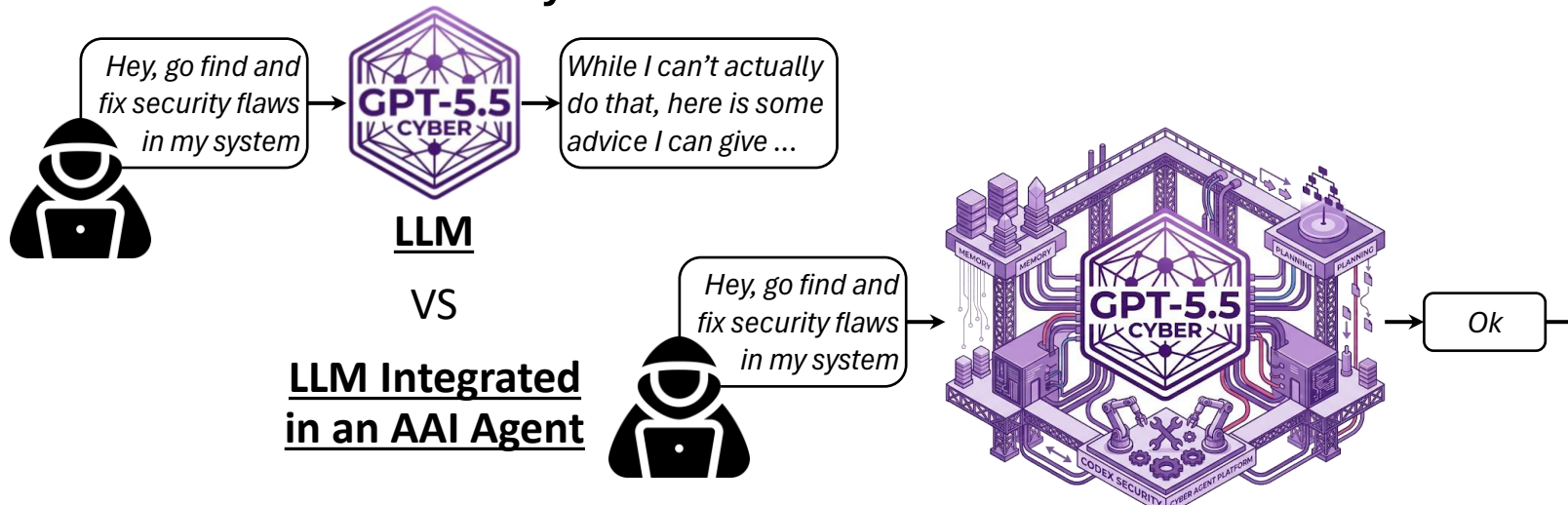


Agentic AI (AAI)

- ▶ Pretrained LLMs have demonstrated a good understanding of textual instructions as is
 - ▶ Due to enormous parameter count and pretraining data.
- ▶ **Consider:** Reddit probably have examples of phishing and benign emails.
 - ▶ If LLMs are pretrained on these sources, they should have learned a “phishing vs benign email” pattern somewhere inside their billions of parameters.
 - ▶ **With good scaffolding around it**, pretrained LLMs can leverage its already-learned patterns to **act** like a finetuned phishing classifier.
- ▶ Agentic AI (AAI) is the framework that allows an LLM to **autonomously observe and interact with an environment** via its generated textual outputs. 

Agentic AI (AAI)

- ▶ AAI is a paradigm shift from traditional ML and DL.
 - ▶ We are revising tasks to what LLMs can understand with their pretrained capabilities instead of adjusting the parameters of an LLM to a task via finetuning!
- ▶ **Scaffold Engineering:** Building the system around an LLM (the AAI Agent) to accomplish tasks an LLM cannot by itself.



OpenAI Expands Daybreak With GPT-5.5-Cyber to Help Defenders Patch Security Flaws

👤 Ravie Lakshmanan 📅 Jun 23, 2026

OpenAI on Monday [said](#) it's releasing an improved version of its GPT-5.5-Cyber model to trusted defenders as part of the [Daybreak initiative](#) the artificial intelligence (AI) company announced last month.

Calling GPT-5.5-Cyber its "strongest model yet for finding and helping patch software vulnerabilities," OpenAI said the model can "sustain deeper analysis across large codebases" to identify security issues, validate them in a controlled environment, and develop and test patches.

In tandem, the tech upstart is releasing an update to the Codex Security plugin to speed up the process of discovering and patching vulnerabilities in existing systems, alongside preventing new vulnerabilities from entering production codebases.

"Developers can run deep scans or review recent changes, generate reports with severity, affected code locations, validation evidence, and remediation guidance, trace attack paths, build threat models, validate findings, and generate codebase-specific patches for review," OpenAI said.

On top of that, the plugin can triage and validate existing findings from scanners, advisories, bug-bounty reports, or ticketing systems, and then facilitate patch generation at scale to quickly close a backlog of vulnerabilities.

Agentic AI (AAI)

- While AI companies are constantly improving LLMs, AAI has significantly expanded what **you** can do with those LLMs.

WSJ WSJ

Here's What OpenClaw Agents Are Doing Today

Developers, executives and startups at Nvidia's GTC event provided a glimpse into how age...

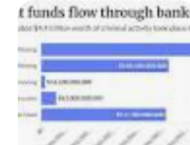


Mar 18, 2026

AB American Banker

How FNBO uses agentic AI to investigate financial crime

The Omaha bank says AI agents halve the time required to research fraud, sanctions violations and...



2 days ago

Cybersecurity Dive

US and allies urge 'careful adoption' of AI agents

New guidance from a coalition of Western governments underscores the difficult-to-predi...



May 1, 2026

Cyber Magazine

Lloyds Deploys Agentic AI to Fight Cyber Fraud in Real Time

Lloyds has prevented over £1bn in fraud, as it scales AI-powered tools and real-time scam detection to...



2 weeks ago

IBM

Transforming biopharma at scale: A new operating model for life sciences with connected data and Agentic AI

Agentic AI-enabled transformation at scale depends on connected data foundations to power redesigned...

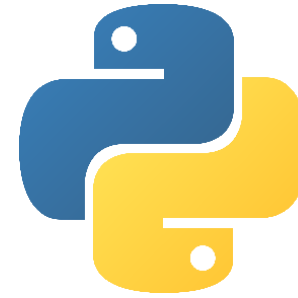


2 days ago

Spoiler for a later week's topic.

Online M.S. in AI

Ongoing Python Example

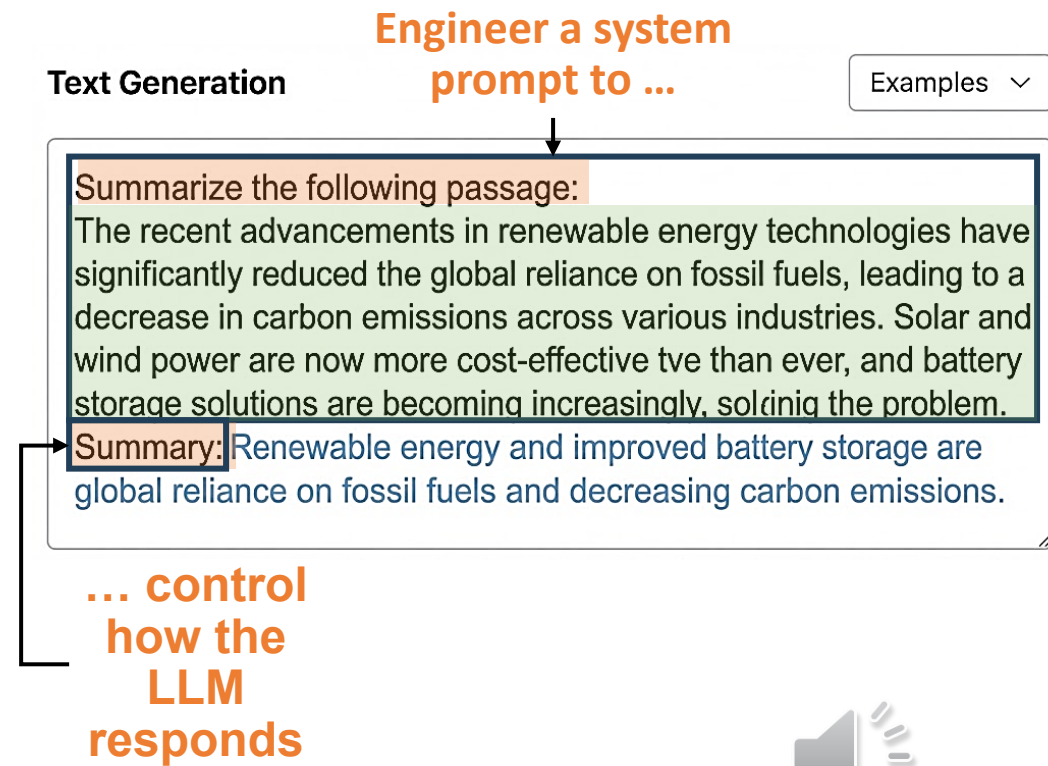


- ▶ **Python example:** Phishing Website Detection with AAI
- ▶ **Data:** Phishing and benign websites
- ▶ **Key Packages:** Request, Anthropic (API), Transformer, OpenClaw
- ▶ Moving forward, we will be using the same python example throughout the entire lesson.
 - ▶ Each section in the Jupyter notebook will correspond to different AAI techniques
 - ▶ The final sections will walkthrough how to setup an OpenClaw-based phishing detector on your system.
- ▶ So, how does one design AAI systems?



Designing AAI Agents

- ▶ **Recall:** LLMs are transformer models, which generates its output based on **every part** of textual input sequence **simultaneously**.
 - ▶ AAI techniques aim to optimally adjust the LLM's input for desired outputs. System prompts are crucial for this.
- ▶ **System Prompt:** Elaborate instruction sets that controls how an LLM responds
 - ▶ Writing good system prompts can control an LLM's output and bypass the need for training (**Prompt Engineering**).
- ▶ **Core AAI Intuition:** System prompts are the “parameters” of AAI systems.

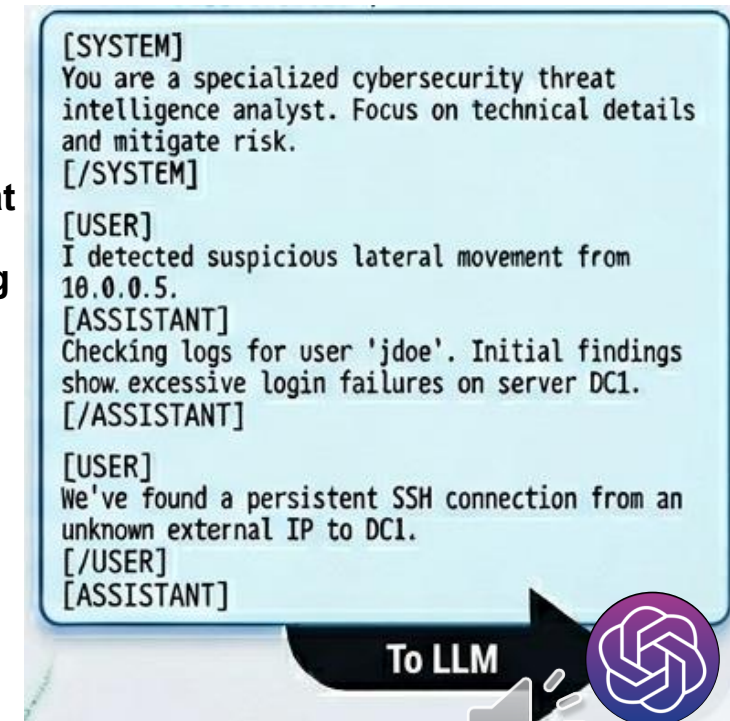
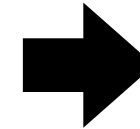


Designing AAI Agents

- ▶ In an AAI framework, LLMs ingest textual inputs of three elements:
 - ▶ **System Prompt**
 - ▶ **Trajectory:** History of the conversation
 - ▶ **New Prompt**
- ▶ The textual input's formatting is handled with Chat Templates.



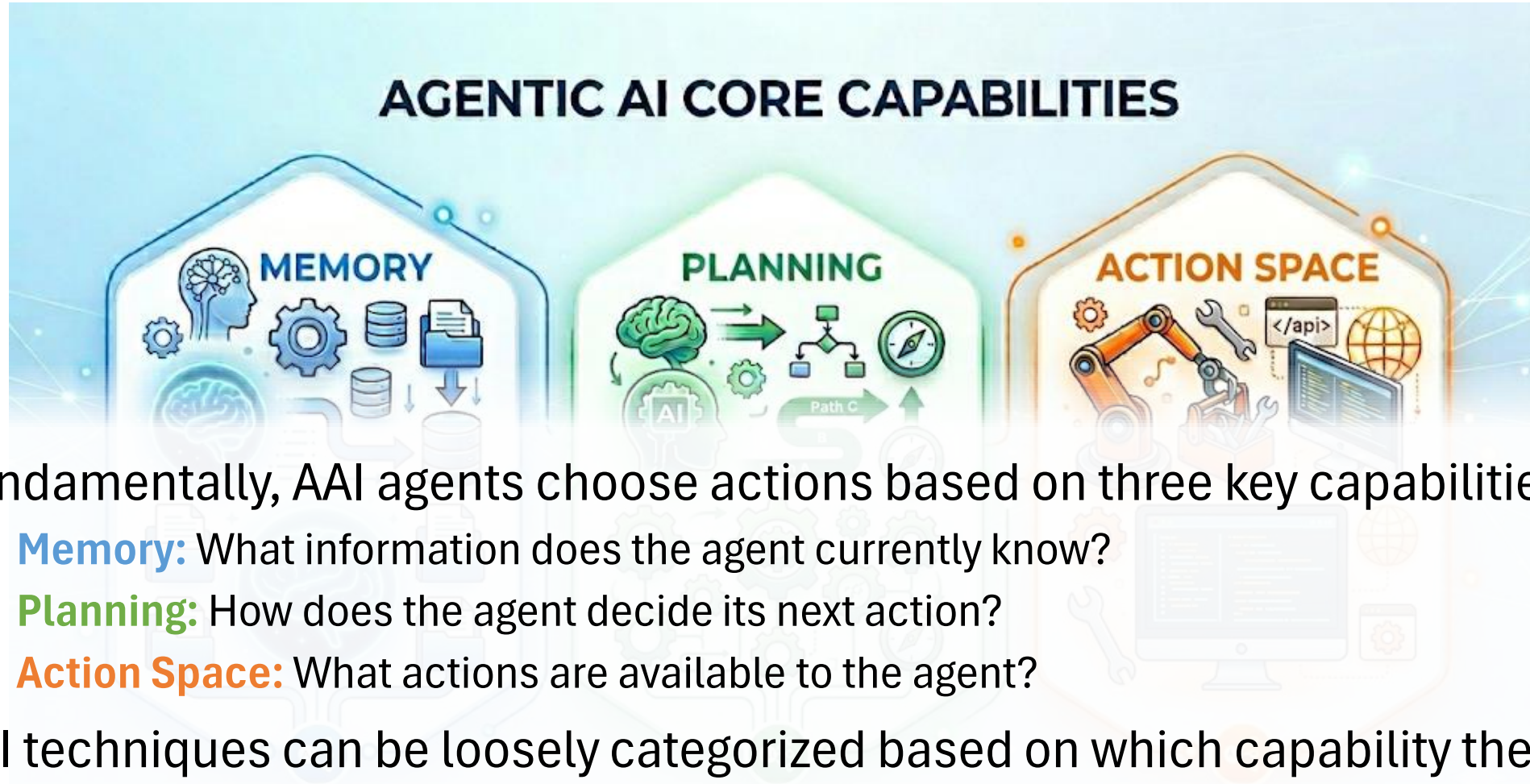
Apply Chat Template formatting



To LLM



Designing and Improving AAI Agents

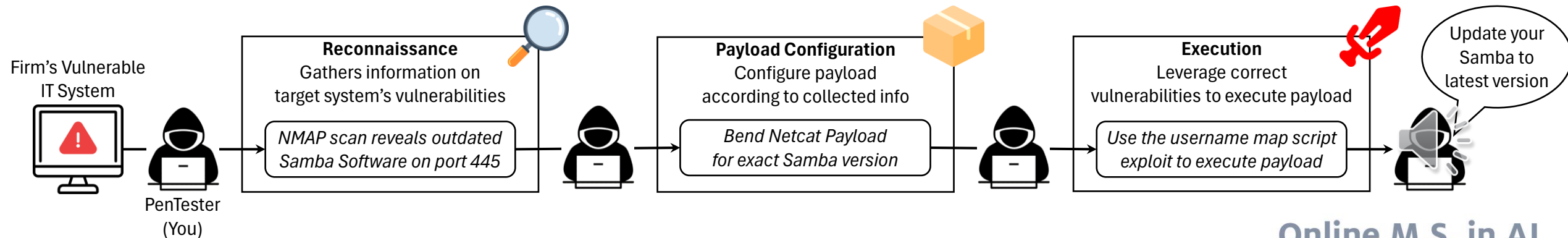


- ▶ Fundamentally, AAI agents choose actions based on three key capabilities:
 - ▶ **Memory:** What information does the agent currently know?
 - ▶ **Planning:** How does the agent decide its next action?
 - ▶ **Action Space:** What actions are available to the agent?
- ▶ AAI techniques can be loosely categorized based on which capability they focus on improving.



Improving Agentic **Memory**: Few-Shot Prompting

- ▶ How do we improve the memory of an AAI agent for a task?
 - ▶ **Recall:** LLMs are already pretrained with general language knowledge in its weights.
- ▶ Suppose you are a senior analyst trained in Penetration Testing (PenTesting).
 - ▶ Firms hire you to find vulnerabilities within their IT systems before adversaries can.
 - ▶ To PenTest, you enter a shell command and interpret (based on your experience and expertise) the CMD results to choose your next shell command.



Improving Agentic **Memory**: Few-Shot Prompting

- ▶ Suppose you now must train a new hire who has some cybersecurity knowledge (i.e., pretrained brain) but no practical PenTesting experience.
 - ▶ How would you improve what the new hire knows (their memory)?
 - ▶ **Consider:** Show them what commands you would choose in a few example situations so they can reference your examples for future PenTesting tasks!
- ▶ **Few-Shot Prompting:** Include specific examples in the system prompt to guide the LLM towards desired output.

Standard
System
Prompt

Few Shot
Examples

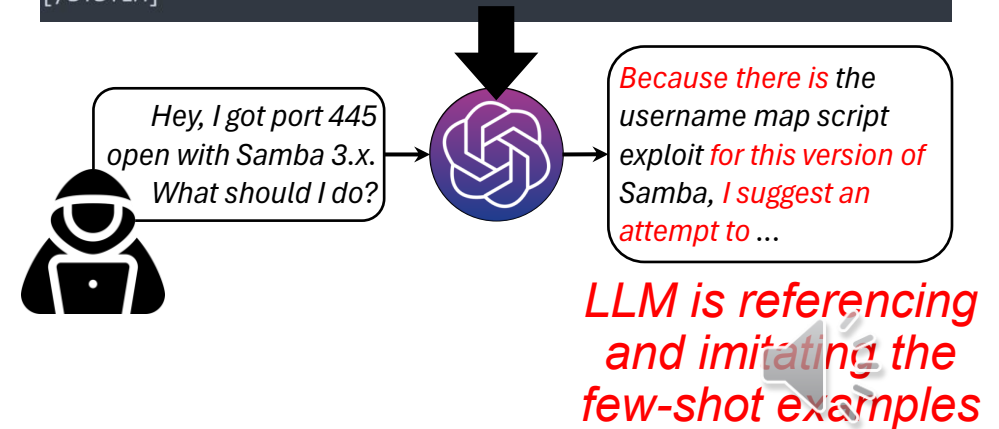
```
[SYSTEM]
You are an automated PenTesting assistant for authorized security audits. PenTesters will describe their situation and you will advise them on their next course of action.

Here are some examples of how you should respond to the PenTester's prompts:

Input: "Port 80 is open with Apache 2.4.49"
Output: "I suggest an attempt to read /etc/passwd using curl."

Input: "Port 22 open with OpenSSH 8.2p1"
Output: "Because there no immediate public exploits for this version of OpenSSH, consider running hydra for common credential spraying."

[/SYSTEM]
```



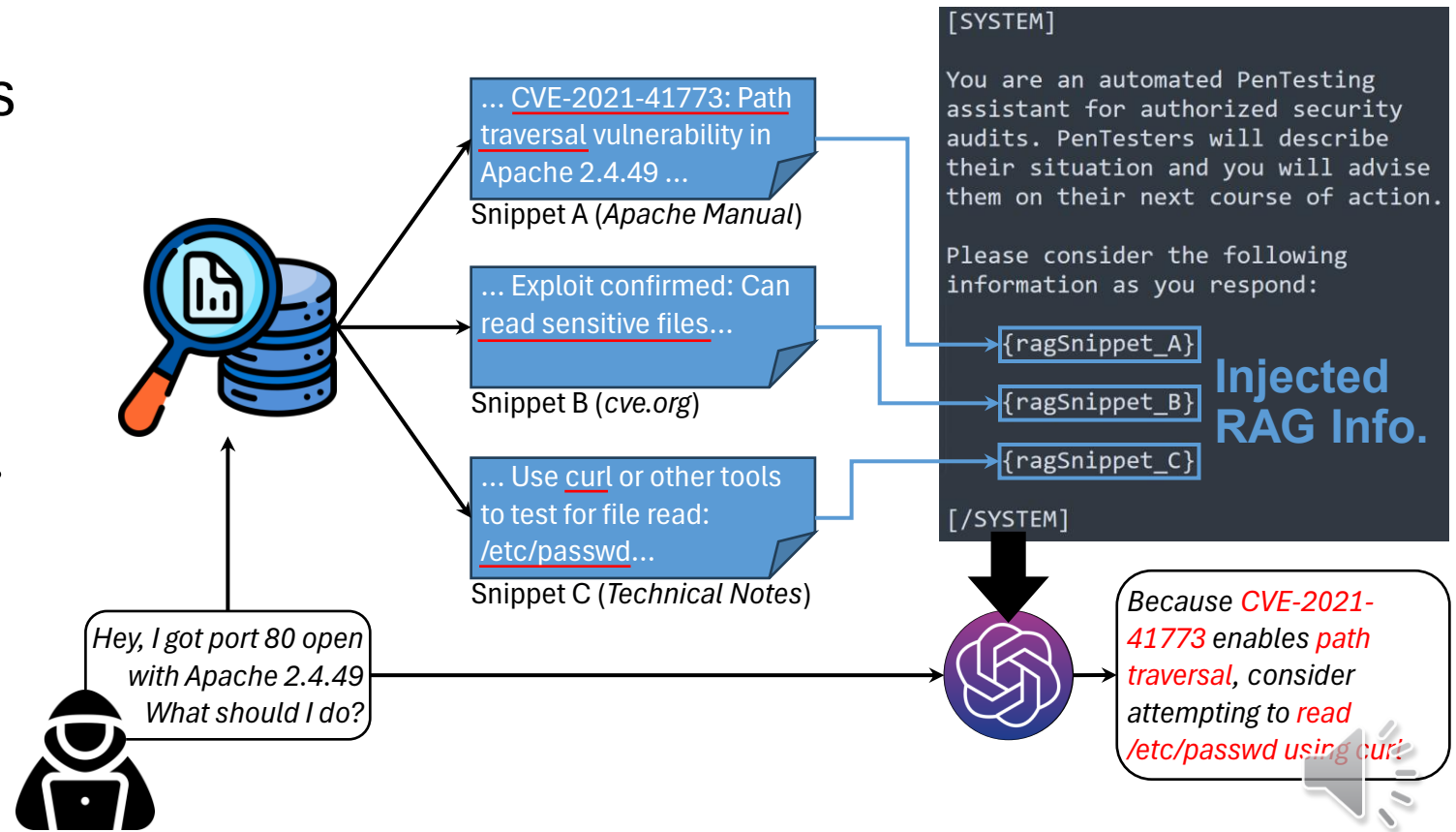
Improving Agentic **Memory**: Retrieval Augmented Generation (RAG)

- ▶ LLMs can be highly dependent on the data used during its pre-training.
 - ▶ If the training data is collected in 2023, it won't know what happened in 2026.
 - ▶ LLM might make up some incorrect facts (hallucination) because their pre-trained memory lacks the relevant information!
- ▶ Suppose you are training your new hire. Would you expect them to already know all the latest discovered vulnerabilities?
 - ▶ Would you even know as a senior PenTester?
 - ▶ **Consider:** It may be more reasonable for you to consult relevant documentation (i.e., retrieve relevant information) as you choose your PenTesting command.



Improving Agentic **Memory**: Retrieval Augmented Generation (RAG)

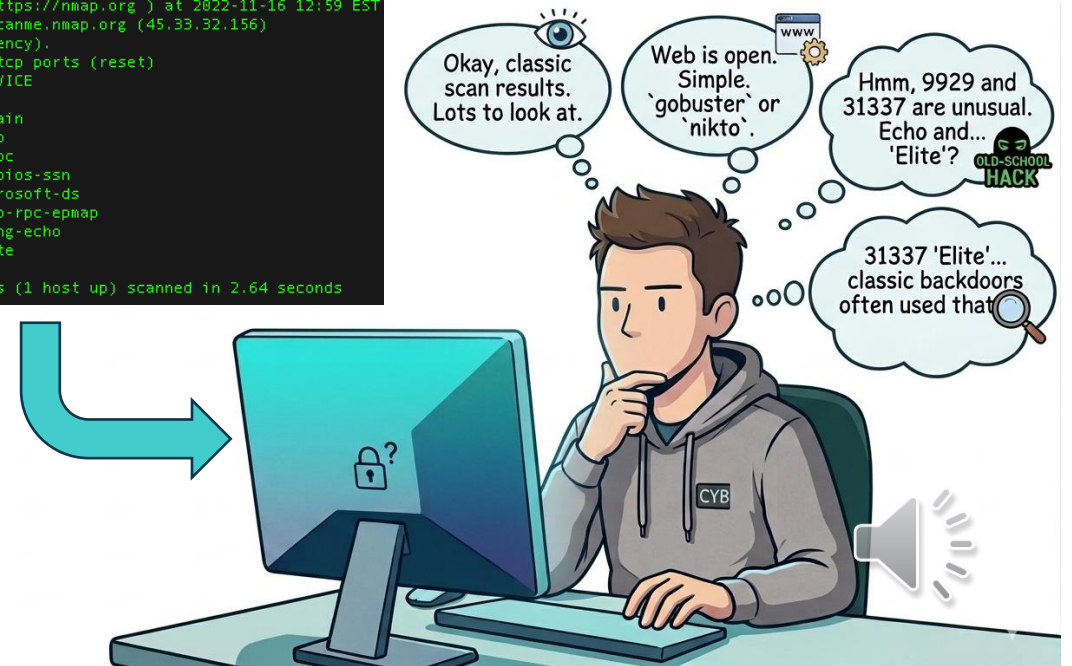
- ▶ **Retrieval Augmented Generation (RAG)** injects retrieved, specific knowledge into the system prompt so that the LLM's outputs are accurate and up-to-date.



Improving Agentic **Planning**: Reasoning

- ▶ As tasks grow in complexity, agentic planning becomes crucial.
 - ▶ Challenging to anticipate every edge case or multi-step pivot in a set system prompt → LLMs must think through challenging problems on their own.
- ▶ Suppose you are a PenTester that just finished a nmap scan of a vulnerable IT system. What is your next step?
 - ▶ You would probably think for a few steps before choosing the next command based on those thoughts.

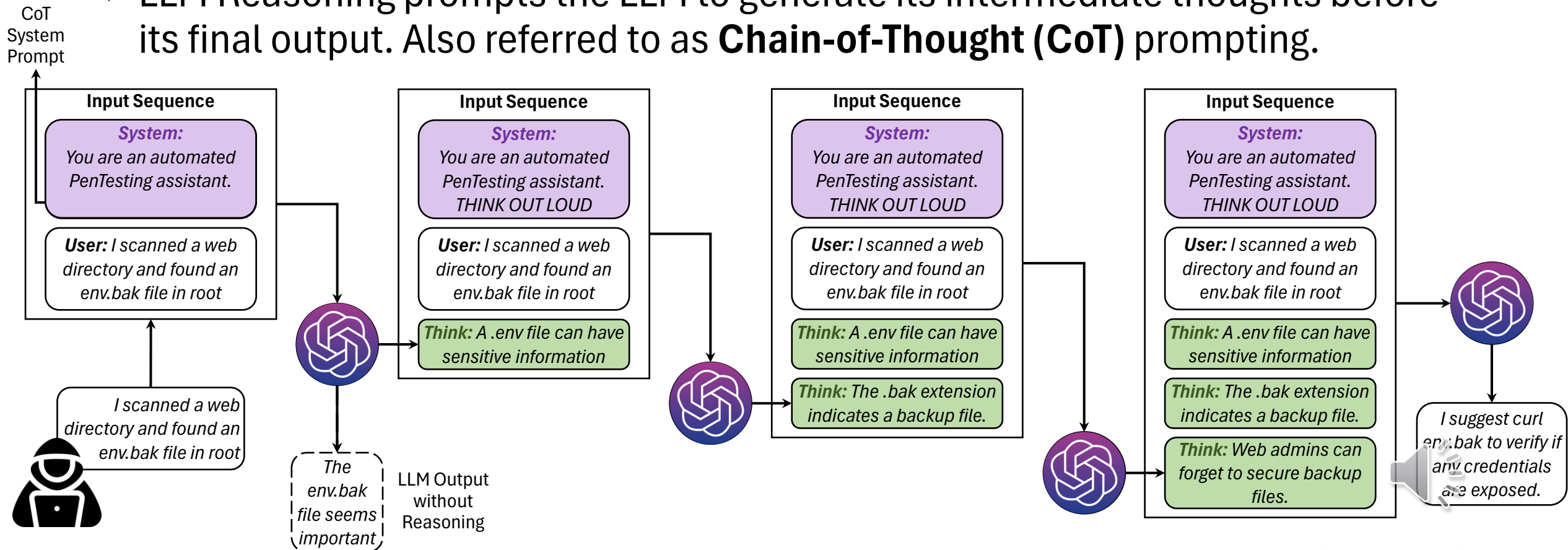
```
[~] ~ sudo nmap -sS scanme.nmap.org
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-16 12:59 EST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.081s latency).
Not shown: 991 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
593/tcp   filtered http-rpc-epmap
9929/tcp  open  nping-echo
31337/tcp open  Elite
Nmap done: 1 IP address (1 host up) scanned in 2.64 seconds
```



Online M.S. in AI

Improving Agentic **Planning**: Reasoning

- LLM Reasoning prompts the LLM to generate its intermediate thoughts before its final output. Also referred to as **Chain-of-Thought (CoT)** prompting.



Improving Agentic **Actions**: Tool Use

- ▶ LLMs primarily interpret and generate text. However, text by itself cannot affect any changes.
 - ▶ **Consider:** A malware's source code remains just text until it is executed by a computer.
- ▶ Cybersecurity often require capabilities outside of text generation. Examples include:
 - ▶ API lookup from malware signature databases.
 - ▶ Executing PenTesting commands for command-line responses.
 - ▶ Saving and opening local files.

This is all harmless text!!

```
struct STUXNET_STRING_STRUCT
{
    void *vTable;           // pointer to table of virtual methods
    void *Buffer;           // pointer to buffer for string
    int Reserbed1;
    int Reserbed2;
    int Reserbed3;
    int Length;             // lengths of the string in buffer
    int MaxLength;          // size of the buffer
};

struct FLAME_STRING_STRUCT
{
    void *vTable;           // pointer to table of virtual methods
    int RefNo;              // object reference counter
    int bInitialized;        // initialization flag
    void *UnicodeBuffer;    // buffer for unicode string
    void *AsciiBuffer;      // buffer for ascii string
    int AsciiLength;         // length of ascii string
    int Reserved1;
    int UnicodeLength;       // length of unicode string
    int LengthMax;           // size of either UnicodeBuffer or AsciiBuffer
};
```



Improving Agentic **Actions**: Tool Use

- ▶ **Tool use** enables LLMs to invoke tools with their generated outputs.
- ▶ Tools comprise of:
 1. Snippets of code that is executable in an environment
 2. Textual description of the tool's capabilities and arguments (Tool Definition).
- ▶ AAI agent decides when to invoke the tool and what arguments to provide.

Tool Description

```
tools = [  
  {  
    "type": "function",  
    "function": {  
      "name": "nmap_scan",  
      "description": "scans the  
        inputted ip address",  
      "parameters": {  
        "type": "object",  
        "properties": {  
          "ipAddr": {  
            "type": "string",  
            "description": "IP of  
              system for scanning,  
              e.g. 127.0.0.1"  
          }  
        },  
        "required": ["ipAddr"]  
      }  
    },  
    ... Descriptions  
      of Other Tools  
  ]
```

Tool Code

```
import subprocess  
  
def nmap_scan(ipAddr):  
    return subprocess.check_output(f"  
        sudo nmap -sS {ipAddr}", shell  
        =True).decode()
```

Example Invocation (Generated by the LLM)

```
{  
  "action": {  
    "tool": "nmap_scan",  
    "toolInput": {  
      "ipAddr": "45.33.32.156 "  
    }  
  }  
}
```

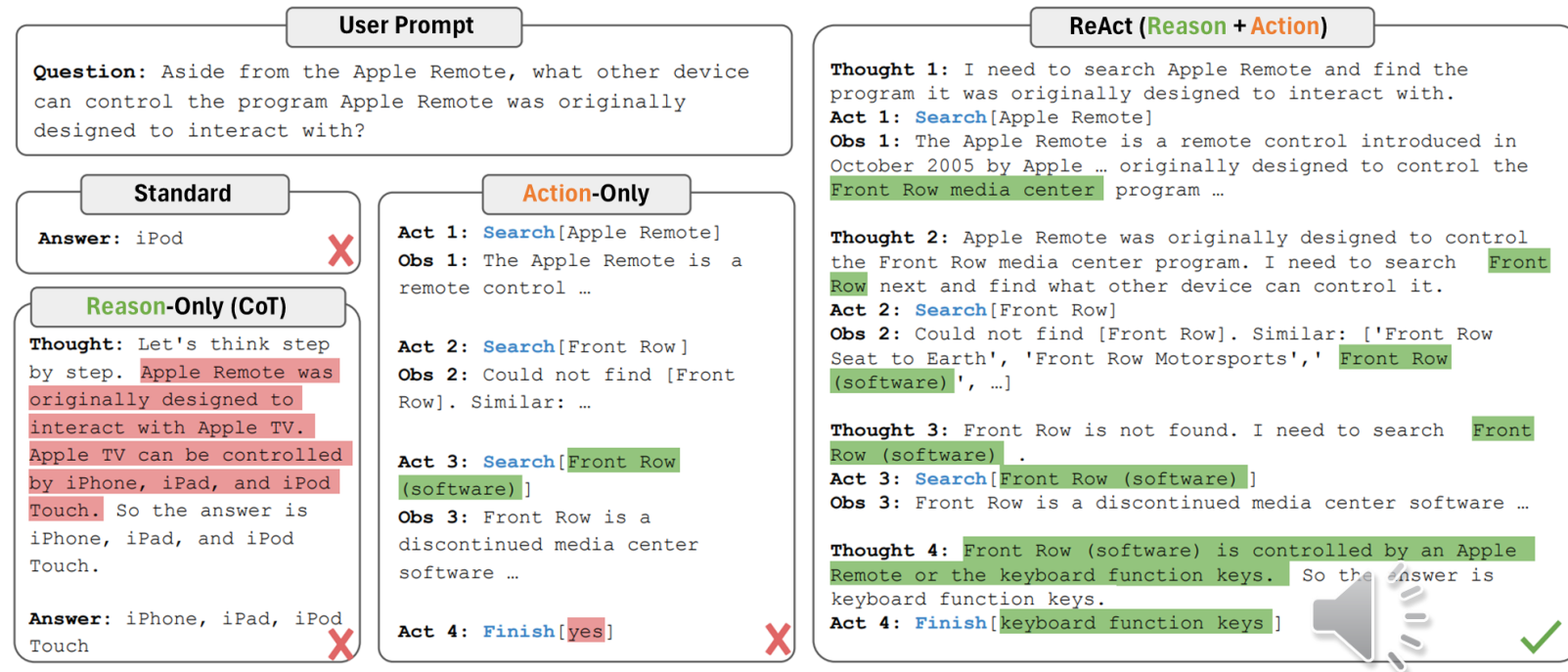


Combining Memory, Planning and Action: ReAct Procedure

- ▶ All three agentic capabilities synthesize into the ReAct (Reasoning + Action) procedure, where LLMs iteratively:

1. Generate thought
2. Invoke a tool
3. Observe the tool's result
4. Add everything to the input sequence
5. Repeat until task completion

- ▶ Most modern AAI agents follow this procedure.



Combining Memory, Planning and Action: ReAct Procedure

Tool Description (Loaded into Agent's **Memory**)

```
tools = [
  {
    "type": "function",
    "function": {
      "name": "nmap_scan",
      "description": "scans the inputted ip address",
      "parameters": {
        "type": "object",
        "properties": {
          "ipAddr": {
            "type": "string",
            "description": "IP of system for scanning, e.g. 127.0.0.1"
          }
        },
        "required": ["ipAddr"]
      }
    }
  },
  ...
]
```

```
import subprocess

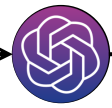
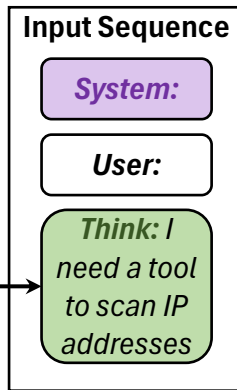
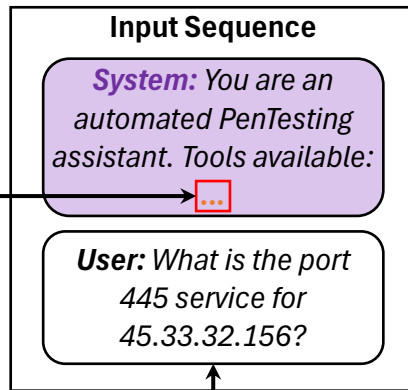
def nmap_scan(ipAddr):
    return subprocess.check_output(f"sudo nmap -sS {ipAddr}", shell=True).decode()
```

Tool Code



What is the port 445 service for 45.33.32.156?

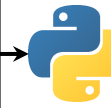
Step 1: Generate Thought



Step 2: Invoke Tool



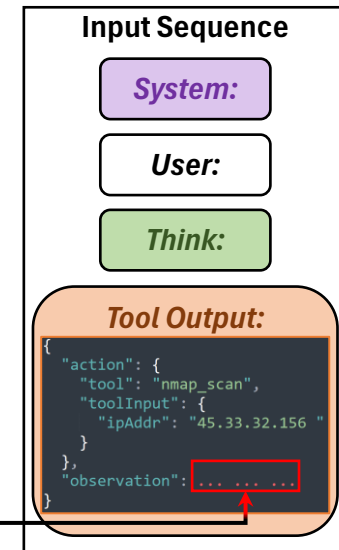
Python environment runs nmap_scan with 45.33.32.156 as input



nmap_scan function result

```
➦ ~ sudo nmap -sS scanme.nmap.org
Starting Nmap 7.93 ( https://nmap.org )
Nmap scan report for scanme.nmap.org
Host is up (0.881s latency).
Not shown: 991 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
593/tcp   filtered http-rpc-epmap
9929/tcp  open  nping-echo
31337/tcp open  Elite
```

Step 3: Observe Results



Step 4: Add to Input

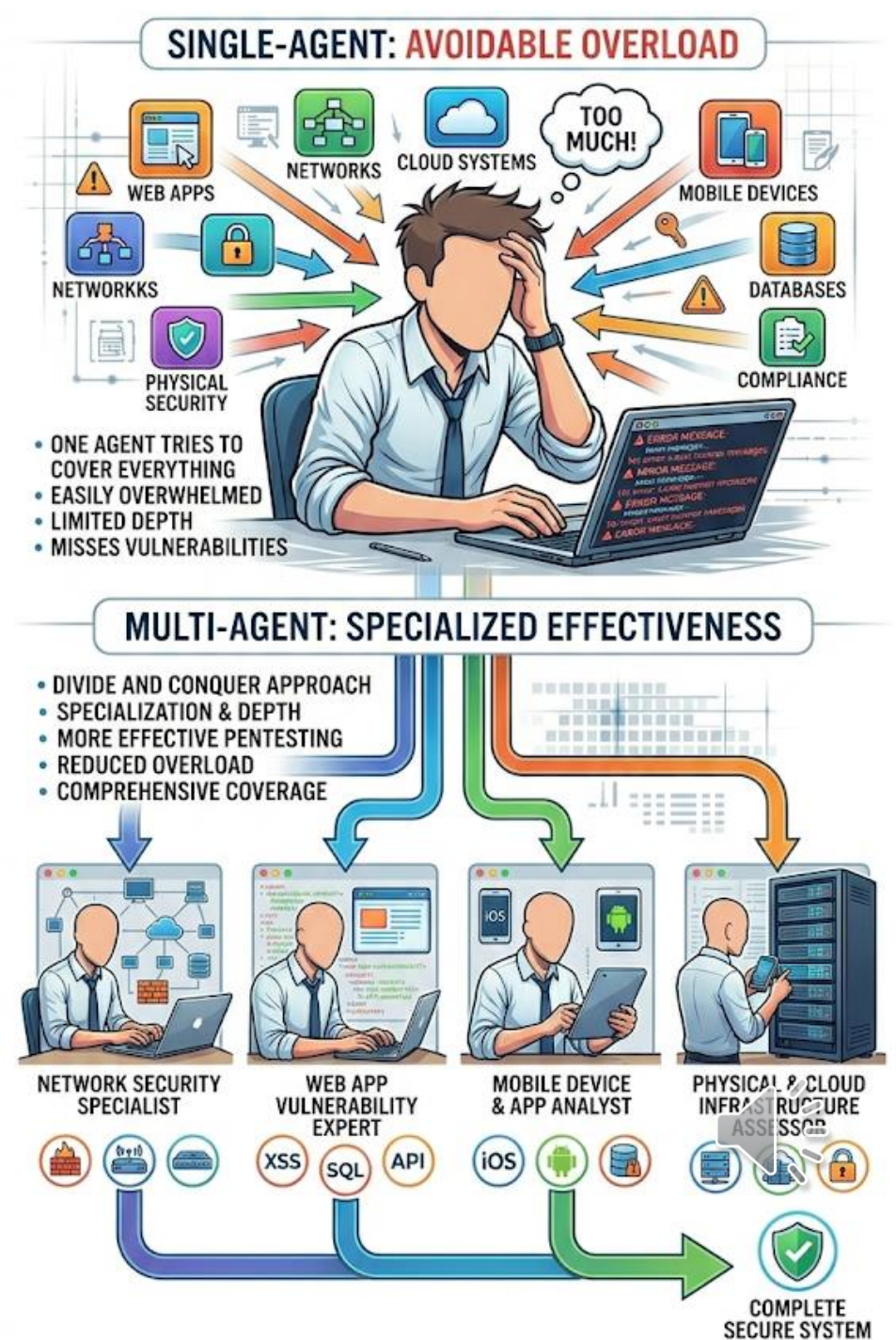


Service is microsoft-ds

Online M.S. in AI

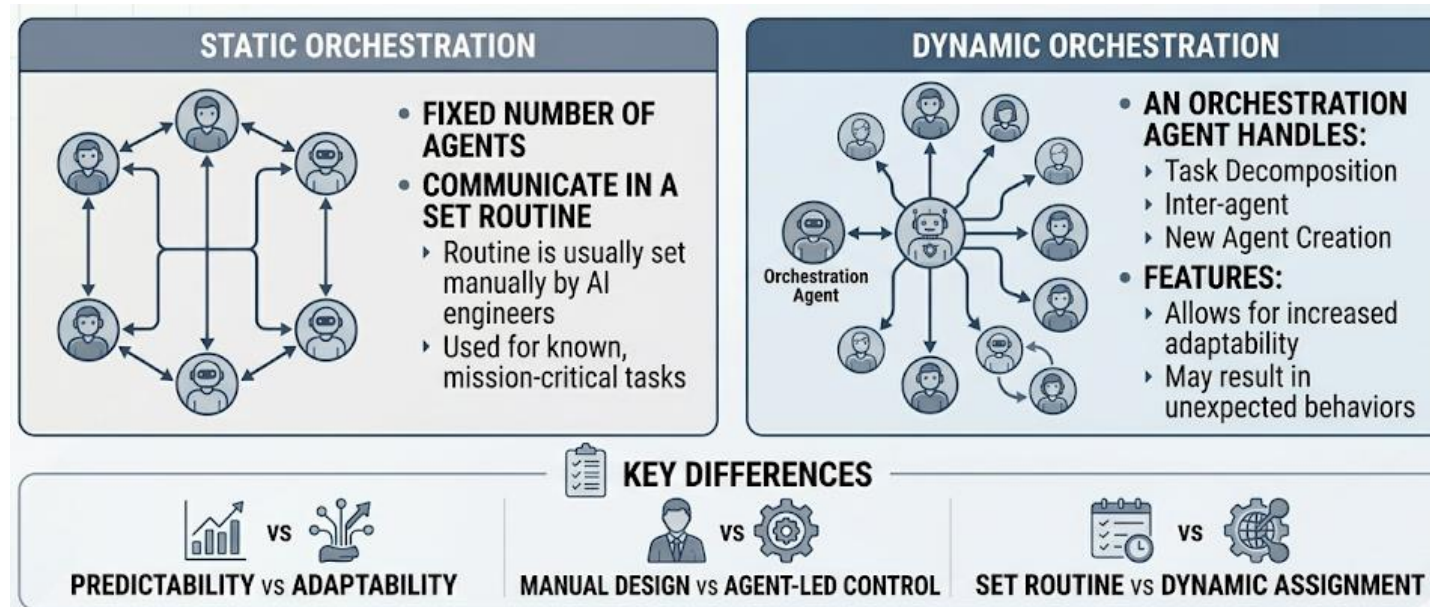
Multi-Agent Systems

- ▶ Single AAI agents can handle tasks that are normally too complicated for standard LLMs.
- ▶ Multi-Agent AAI systems can handle an even more complicated task by decomposing it into sub-tasks and assigning each to specialized agents.
 - ▶ Reduces the cognitive load for individual agents, resulting in better performance.
- ▶ **Consider:** Most PenTesters specialize in certain areas and test for specific types of vulnerabilities.



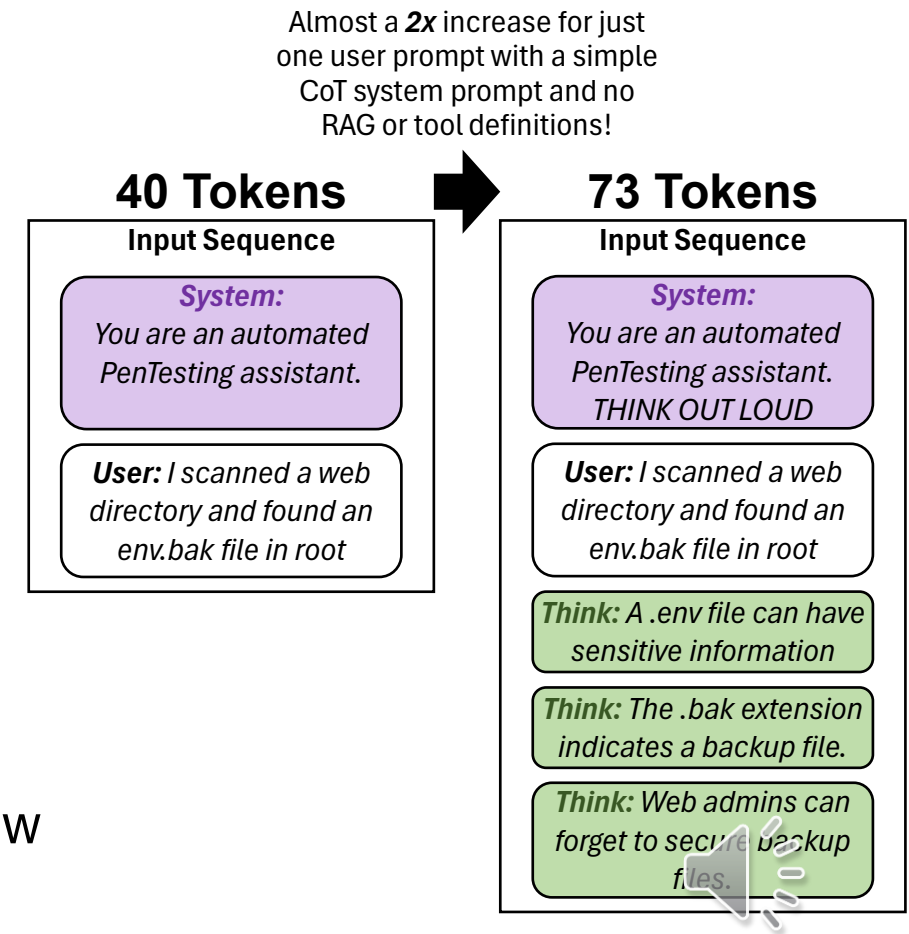
Multi-Agent Systems

- Multi-Agent AAI systems can be categorized based on how their different agents interact with each other (Orchestration):
 - Static Orchestration:** Fixed number of agents that communicate in a set routine.
 - Dynamic Orchestration:** An orchestration agent handles task decomposition, inter-agent communications, and new agent creation.



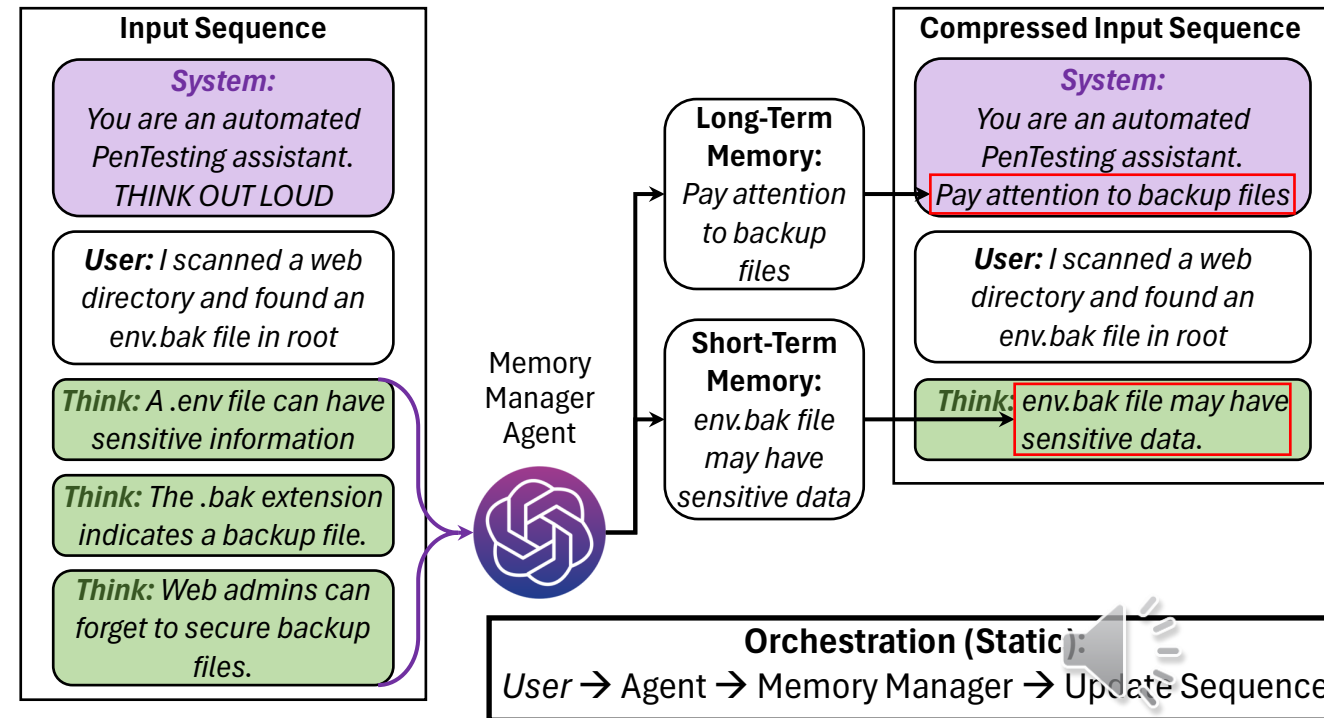
Static Orchestration: Context Management

- ▶ With few-shot examples, RAG injections, reasoning thoughts, and tool definitions, an agent's input sequence can rapidly increase.
 - ▶ Size of input sequences can reach thousands of tokens, especially for multi-turn conversations.
- ▶ LLMs have fixed context windows and cannot ingest sequences that are longer than it.
 - ▶ ChatGPT-5.5 has a ~400K token context window
 - ▶ Claude Opus 4.6 has a ~1 million token context window
 - ▶ Gemini 2.5 Pro has a ~2 million token context window



Static Orchestration: Context Management

- ▶ To save on sequence length, AAI agents employ another agent as a memory manager to compress the input sequence.
- ▶ Memory managers summarize long input sequences into two types of memory:
 - ▶ **Long-Term Memory:** Knowledge that should be incorporated into how the agent approach future problems.
 - ▶ **Short-Term Memory:** Knowledge specific to this conversation.



References:

- Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J. and Zhang, Y., 2026. A-mem: Agentic memory for llm agents. Advances in Neural Information Processing Systems, 38, pp.17577-17604.
- Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. and Yao, S., 2023. Reflexion: Language agents with verbal reinforcement learning. Advances in neural information processing systems, 36, pp.8634-8652.

Static Orchestration: Automatic Prompt Refinement



- ▶ AAI systems need to respond and adapt to complex environments autonomously.
 - ▶ Would you like your new hire to ask you questions for every new PenTesting situation they find themselves in?
 - ▶ Instead, the new hire should learn from his mistakes and update his approach to PenTesting accordingly on his own!
- ▶ **Recall:** All AI models optimize by autonomously adjusting their parameters to minimize a loss function (mistake) via mathematical algorithms.
 - ▶ However, AAI's "parameters" are its system prompts, which are textual. AI models have numeric parameters (ANN's weights, K-mean's centroid coordinates, etc.).

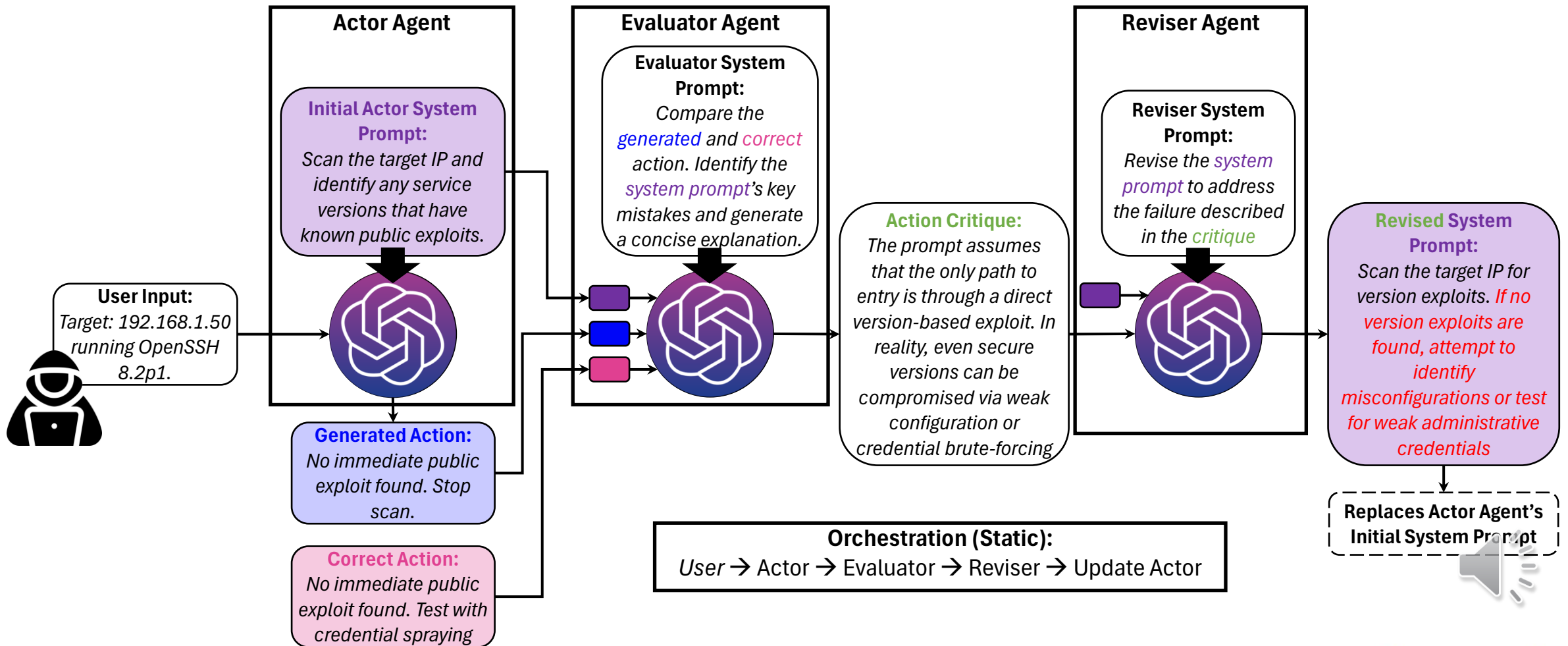


Static Orchestration: Automatic Prompt Refinement

- ▶ How can we automatically adjust system prompts to optimize an agent's outputs (also textual)?
 - ▶ Fortunately, we have a mathematical algorithm that inputs and outputs textual data: other agents!
- ▶ **Automatic Prompt Refinement (APR)** leverages a multi-agent system to improve one agent's outputs by refining its system prompts with other agents.
- ▶ APR defines three agents:
 - ▶ **Actor Agent:** Generates action based on user input and is the agent being optimized.
 - ▶ **Evaluator Agent:** Critiques the Actor Agent's system prompt based on the action's mistake.
 - ▶ **Reviser Agent:** Rewrites the Actor Agents's system prompt according to the Evaluator Agent's critique.

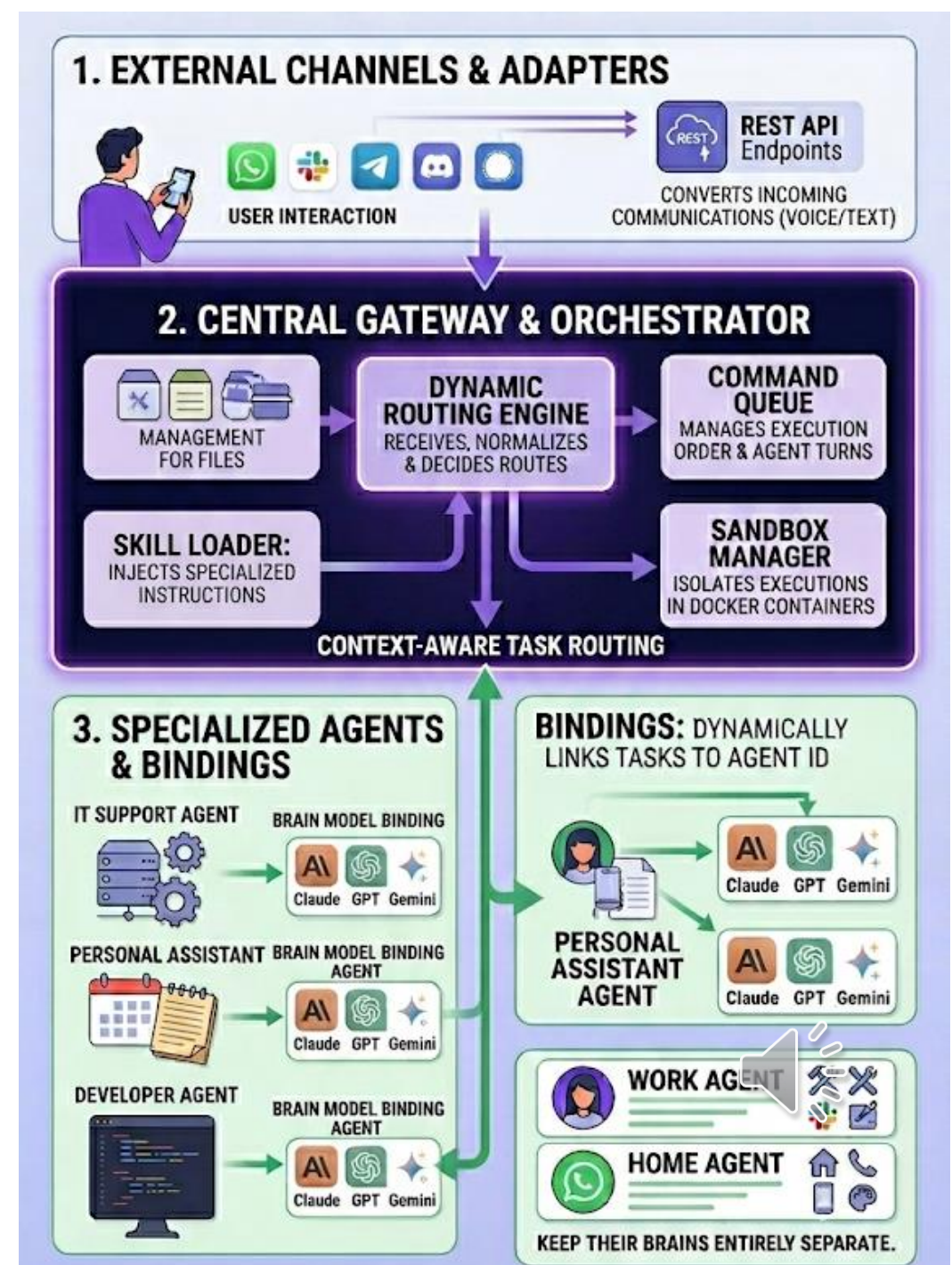


Static Orchestration: Automatic Prompt Refinement



Dynamic Orchestration: OpenClaw

- Dynamic Orchestration represent state-of-the-art AAI systems.
 - Exact orchestration mechanisms are an active area of AAI research and outside the scope of this class.
- For this class, we will focus on one of the most prevalent AAI system with dynamic orchestration: OpenClaw.
- OpenClaw is a multi-agent AAI framework designed around continual operation.
 - Many capabilities discussed are exhibited via Markdown files (.md) that OpenClaw can generate, edit, and delete as it runs.



Dynamic Orchestration: OpenClaw

▶ Significant OpenClaw Markdown files:

▶ Memory:

- ▶ USER.md: Stores context about the human user, including preferences and goals.
- ▶ MEMORY.md: Stores persistent memory, patterns, and important facts.
- ▶ IDENTITY.md: Specifies the agent's role, title, and organization
- ▶ skill.md(s): Specialized sets of instructions for certain tasks

▶ Planning:

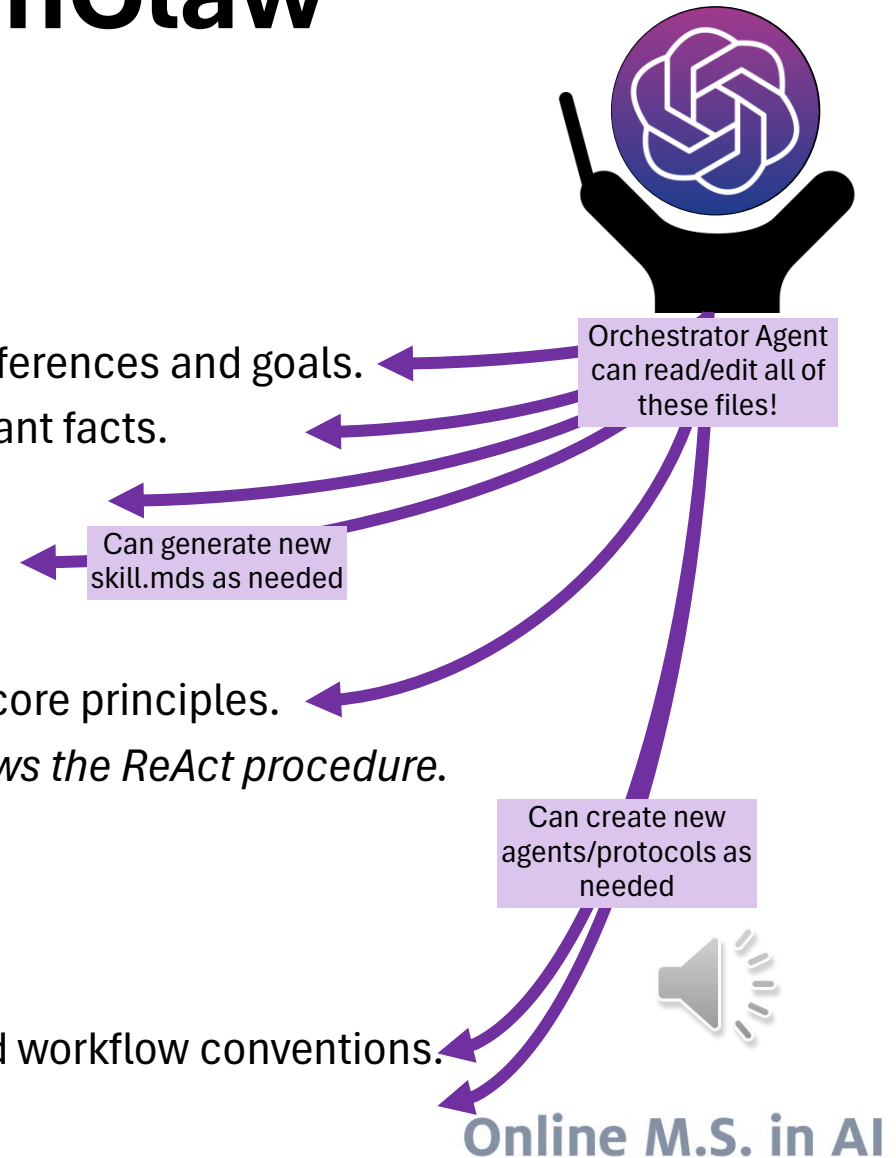
- ▶ SOUL.md: Defines the agent's personality, beliefs, voice, and core principles.
- ▶ *OpenClaw requires LLMs with reasoning capabilities and follows the ReAct procedure.*

▶ Action Space:

- ▶ TOOLS.md: Documents tools the agent can access.

▶ Dynamic Orchestration:

- ▶ AGENTS.md: Contains operational rules, security policies, and workflow conventions.
- ▶ HEARTBEAT.md: Defines scheduled, recurring tasks.

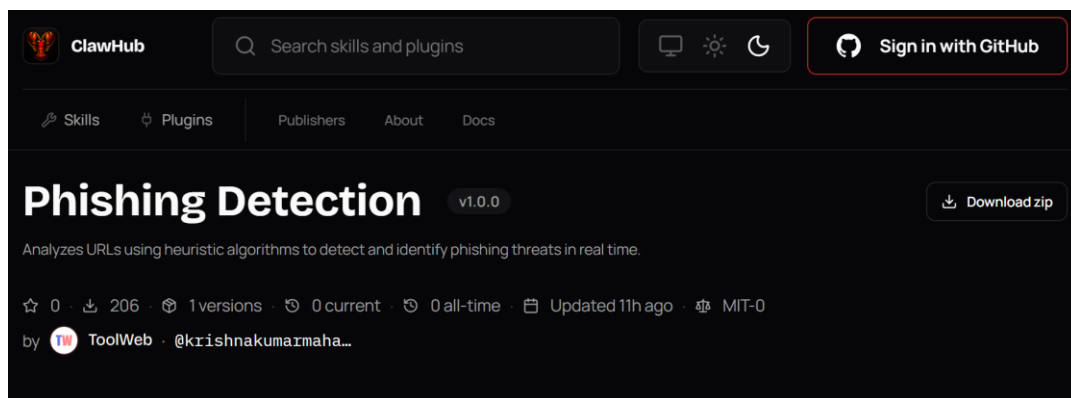
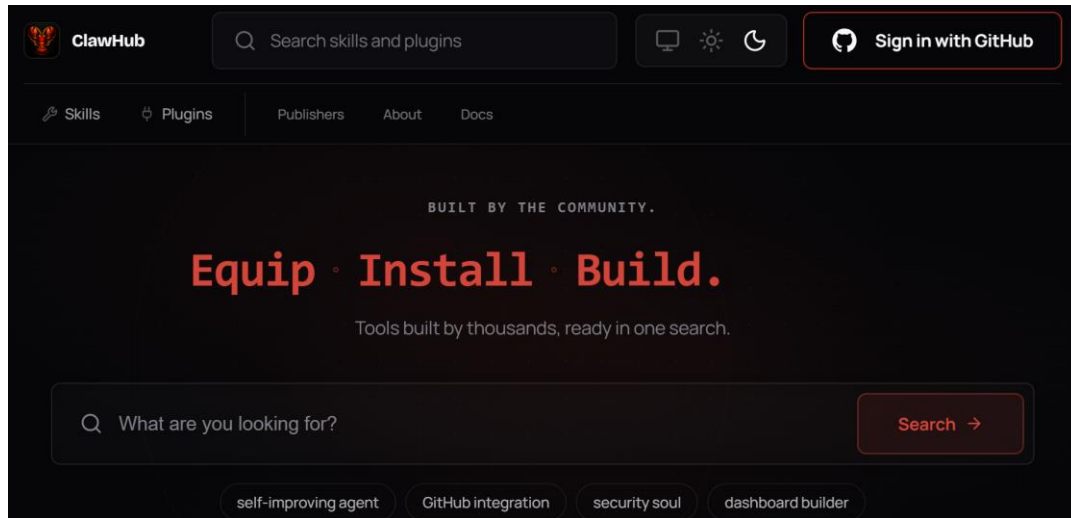


Open-Source Agents

- ▶ A key aspect of OpenClaw agents is skills, or markdown files that contain a system prompt for certain tasks.
- ▶ Skills are a mixture between Tool Use and RAG prompts.
 - ▶ Each skill has a short description that the Orchestrator Agent parses.
 - ▶ Depending on the situation, the Orchestrator Agent will inject the skill into an agent's system prompt, like RAG.
- ▶ While the Orchestrator Agent can generate new skills, humans still provide the highest quality skills.
 - ▶ These skills can easily be shared and integrated into your OpenClaw systems via clawhub.ai

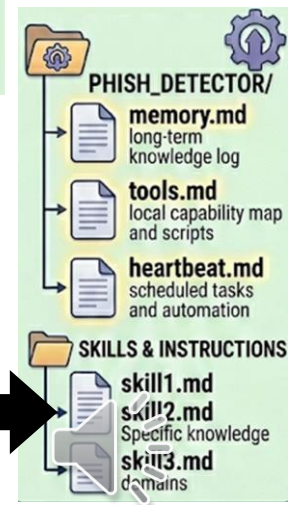
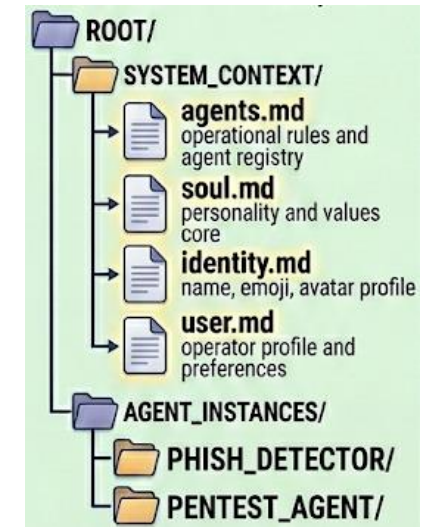


Open-Source Agents



HEURISTIC_DETECTION.md

```
1 ---
2 name: Heuristic Phishing Detection
3 description: Analyzes URLs using
4               heuristic algorithms to detect and
5               identify phishing threats in real
6               time.
7 ---
8 # Overview
9
10 Heuristic Phishing Detection is a
11 security-focused API that analyzes
12 suspicious URLs to identify phishing
13 attacks and malicious links. Using
14 advanced heuristic algorithms, the
15 tool examines URL structure, domain
16 characteristics, and behavioral
17 patterns to determine threat
18 likelihood without relying solely on
19 blocklists.
```



Online M.S. in AI

Open-Source Agents

- ▶ Clawhub.ai also shares plugins, or standardized libraries for AAI tool use.
 - ▶ Sharing skills and plugins significantly lowers the difficulty of implementing OpenClaw-based systems.
- ▶ Example personal use cases:
 - ▶ Notes manager
 - ▶ File organizer
 - ▶ Etc.
- ▶ Example cybersecurity use cases:
 - ▶ Risk + vulnerability assessment
 - ▶ Generalized OSINT web crawler
 - ▶ Safeguarding against misalignment attacks



CIO JOURNAL

Here's What OpenClaw Agents Are Doing Today

Developers, executives and startups at Nvidia's GTC event provided a glimpse into how agent frameworks, built on orchestration platform OpenClaw, can be used

By Isabelle Bousquette

March 18, 2026 4:54 pm ET

Aa Listen (1 min)

Nader Khalil, Nvidia's director of developer tech, and Luke Wignall, director of technical product marketing, at the Build-a-Claw tent at Nvidia's GTC event in San Jose, Calif. ISABELLE BOUSQUETTE/WSJ

SAN JOSE, Calif.—The energy was electric inside a tent at Nvidia's annual GTC event, where developers and tech executives clamored around high-top tables to experiment with OpenClaw.

The open-source AI agent orchestration system, which went viral in recent weeks, became a key topic of Nvidia's event. On Monday, CEO Jensen Huang announced NemoClaw, a software development tool kit aimed at making these OpenClaw agents enterprise-ready.

"The importance is profound. It's the most popular open-source project in the history of humanity," he said.

At Nvidia's on-site Build-a-Claw experience, technologists were building "claws" for everything from road trip planning to job applications, research and content marketing. Nvidia executives were standing by to help.

Claws are autonomous agents and can plan and execute tasks on their own.

Online M.S. in AI

Open-Source Agents (Examples)

ClawHub

Skills 66.4K

All categories

AllTrendingTopMost starredFeatured

SKILL	POPULARITY
Sonoscli @steipete	☆ 56 ↓ 85.1k
<u>Control Sonos speakers</u> (discover/status/play/volume/group).	
Gog @steipete	☆ 933 ↓ 187k
Google Workspace CLI for <u>Gmail, Calendar, Drive, Contacts, Sheets, and Docs</u> .	
Github @steipete	☆ 645 ↓ 192k
<u>Interact with GitHub</u> using the `gh` CLI. Use `gh issue`, `gh pr`, `gh run`, ...	
Weather @steipete	☆ 421 ↓ 162k
Get current weather and forecasts (no API key required).	
Openai Whisper @steipete	☆ 329 ↓ 85.2k
Local <u>speech-to-text</u> with the Whisper CLI (no API key).	

ClawHub

Plugins 1.6K

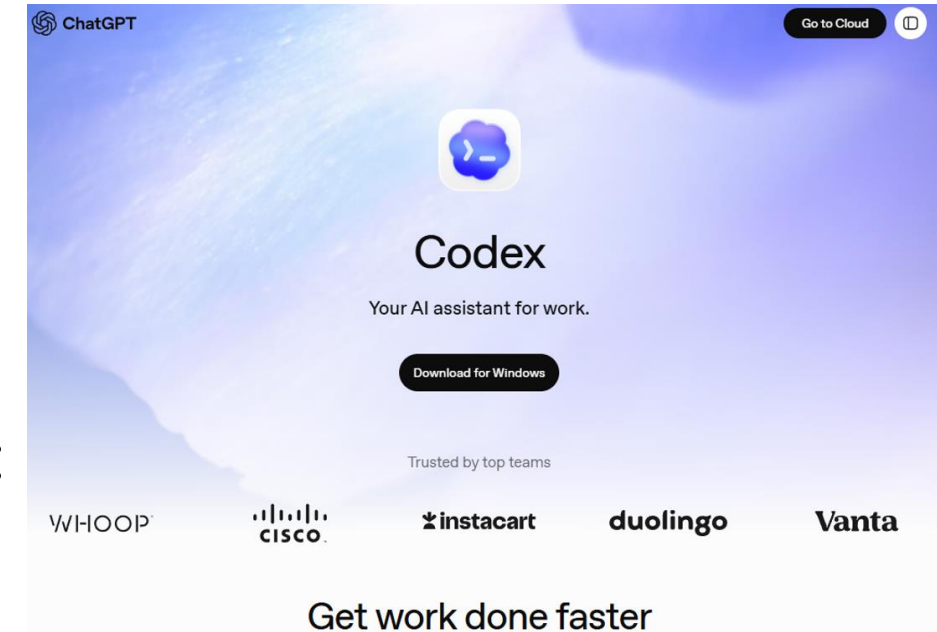
All categories

AllTrendingVerifiedUpdated

PLUGIN	POPULARITY
WhatsApp @openclaw	↓ 148k
<u>OpenClaw WhatsApp channel plugin</u> for WhatsApp Web chats.	
Matrix @openclaw	↓ 38.1k
OpenClaw Matrix channel plugin for rooms and <u>direct messages</u> .	
OpenClaw DeepSeek Provider @openclaw	↓ 485
OpenClaw DeepSeek provider plugin.	
Codex @openclaw	↓ 32.1k
OpenClaw Codex app-server harness and model provider plugin with a...	
OpenClaw Parallel Plugin @openclaw	↓ 373
OpenClaw Parallel <u>web search</u> plugin.	

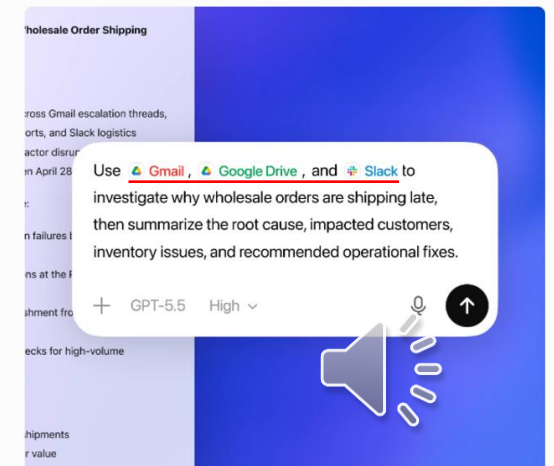
Closed-Source Agents

- ▶ While OpenClaw is impressive, state-of-the-art capabilities still lie with closed-source agents.
- ▶ Major AI providers also offer prebuilt AAI services:
 - ▶ Google's Gemini can access/interact with your drive files via RAG and Tool Use.
 - ▶ OpenAI's Codex is a general-purpose agent interfaced with non-OpenAI tools.
 - ▶ Anthropic's Mythos is a pentesting agent good enough to be banned as a national security threat



Your research assistant

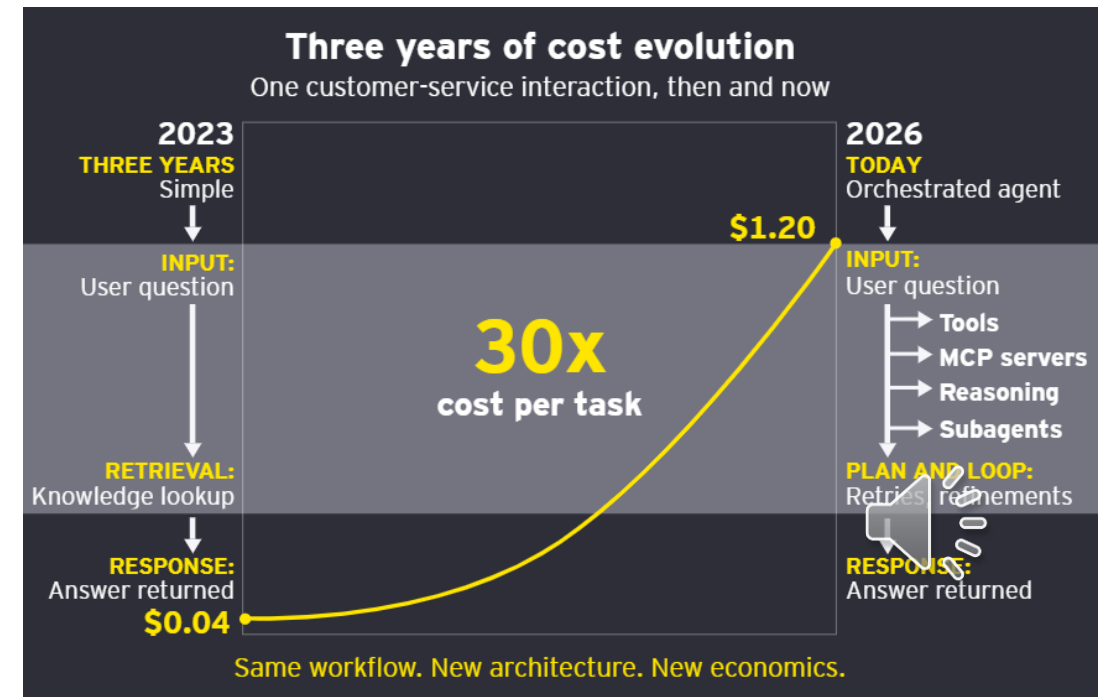
Give Codex the files, notes, data, decisions, and code behind the task. It works from the actual materials that shape the job, so you do not have to flatten real work into a single prompt.



Online M.S. in AI

Mixed-Source Agents

- ▶ Using closed-source agents could be expensive due to token costs from larger and larger input sequences.
 - ▶ Even open-source agents with closed-source LLM backends could get expensive fast
- ▶ Many AAI cybersecurity systems leverage a mixture of closed-source premium LLMs for difficult tasks and open-source local models for simpler tasks.
 - ▶ Choosing which LLM to use for which tasks requires an AI4Cyber analyst who understands the capabilities of each LLM (AI) and the nature of the task itself (cybersecurity).





Review Questions

- ▶ What enables AAI-based systems to bypass finetuning for task adaptation?
- ▶ What are the three central capabilities of an AAI agent? How do all three interact with each other in the ReAct procedure?
- ▶ What are the two mission-critical tasks typically assigned to static orchestration systems?
- ▶ How is OpenClaw designed for continual operation? How does this design focus lead to open-source agents?



Next Week

