BLACK KITE

auxis
A Grant Thornton US company

A Black Kite Data Research Paper

Agentic AI in Cybersecurity:

Use Cases, Variants, and Real-World Fit

Introduction: Why Agentic AI?

In the ever-evolving world of artificial intelligence, a new player has stepped into the spotlight — *Agentic AI.* Unlike traditional models that simply react to prompts, Agentic AI systems are designed to take the **initiative, make decisions, and pursue goals autonomously.** In other words, they don't just follow instructions — *they take the wheel.*

Acting as intelligent agents, these systems can plan, adapt, and even negotiate on our behalf, opening the door to unprecedented possibilities in automation, creativity, and problem-solving. **Agentic AI doesn't wait for permission to act.**

In this whitepaper, we dive into the defining traits of Agentic AI and examine the emerging protocols that enable autonomous, goal-driven behavior. Through the lens of **cybersecurity and risk intelligence,** we highlight how these systems can transform the orchestrate intelligent automation, strategic decision-making, and scalable threat analysis.

Key Concepts & Handy Tools to Know (Before You Dive In)

Workflows

Predefined sequences of tasks or rules executed in a fixed order, often requiring human setup and intervention. Workflows follow scripted paths, where agents act autonomously.

Agentic AI

Systems that can autonomously plan, decide, and act based on high-level goals, adapting as needed. Capable of reasoning and execution.

MCP

Provides language models with contextual information—such as user goals, memory, rules, and tool access.

Agent Swarm

A group of AI agents that collaborate in a decentralized manner to solve complex problems, inspired by swarm intelligence in nature (like ants or bees).

AI Scaffolding

Architectural structures built around LLMs to help it perform complex tasks reliably and efficiently. Includes tools, memory modules, task planners, control logic, and context management systems.

Agent Tools

External functions or APIs that extends its capabilities beyond language processing—such as searching the web, querying a database, or executing code.

LangGraph

A framework for building stateful, multi-agent applications using LLMs, where interactions are structured as graphs of nodes (agents or functions).

CrewAI

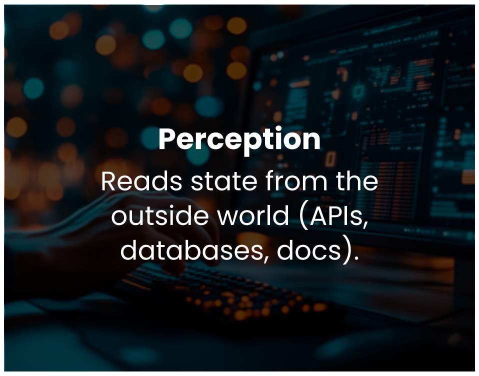
A framework that enables collaboration among multiple specialized AI agents("crew members") through structured teamwork. It assigns roles like project manager, researcher, or writer to each agent.

Table of Contents

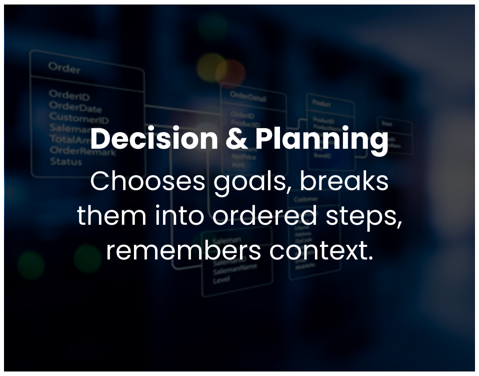
Why Agents Are More Than Fancy Prompts

What Exactly is an AI Agent?

An AI agent is a self-contained micro-service that happens to be powered by an LLM. It bundles three capabilities you don't get from a naked prompt:



Perception
Reads state from the outside world (APIs, databases, docs).



Decision & Planning
Chooses goals, breaks them into ordered steps, remembers context.

Action
Executes steps, evaluates results, loops until goals satisfied.

Because an agent owns the loop **(observe → decide → act)**
it behaves more like a software actor than a chat completion.

From LLM Prompts to Agent Protocols

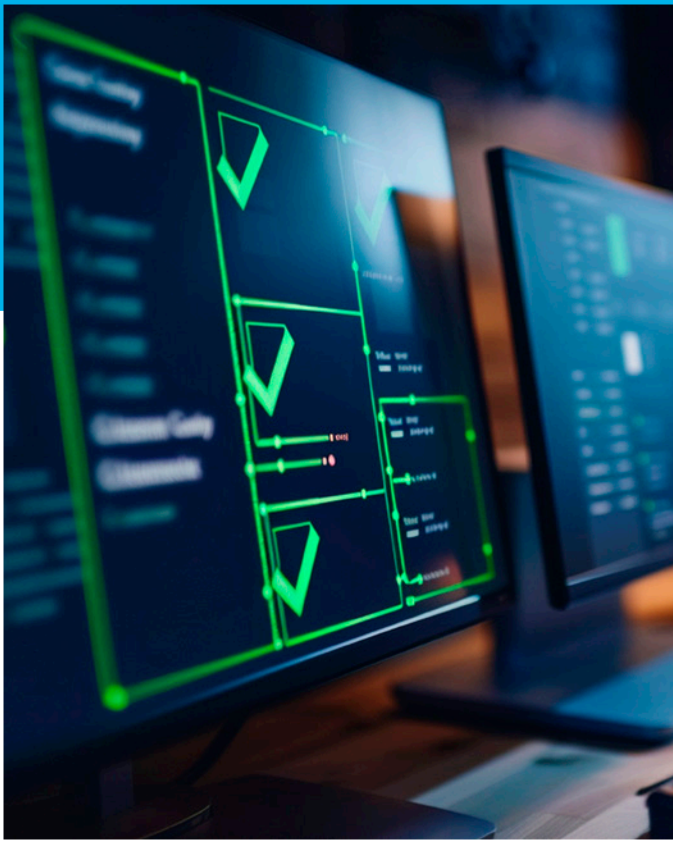
"A prompt is a single instruction; an agent is a project manager."

Capability	Prompts	Agents
Long-term memory & context	X	✓
Multi-step planning	Limited (chain-of-thought)	Native task graphs
Tool / API calling	Manual via function-calling	Declarative in manifest
Collaboration (A2A)	X	Standardized message bus
Governance hooks	Post-hoc log review	Embedded audit trail

Agents expose these features through Model-Context Protocols (MCP) and Agent-to-Agent (A2A) envelopes, letting multiple LLMs (Gemini, GPT-4o, etc.) interoperate safely.

AI Agents and Protocols: The Hidden Architecture Behind Intelligent Automation in Cybersecurity and Risk Management

When Intelligent Machines Need to Talk



Imagine you're at a massive international airport where thousands of planes need to coordinate landings, takeoffs, and gate assignments. Without clear protocols—standardized ways of communication—chaos would ensue.

The same principle applies to AI agents working in complex environments like Third-Party Risk Management (TPRM) and Cyber Threat Intelligence (CTI).

AI agents aren't just chatbots or simple automation tools. They're sophisticated software entities that can perceive their environment, make decisions, and take actions to achieve specific goals. But here's the catch: they need protocols to work effectively, especially when multiple agents collaborate or when they interact with various systems and databases.

What Are AI Agent Protocols?

At their core, AI agent protocols are standardized methods of communication and interaction that enable agents to:

- **Exchange information** with other agents or systems
- **Coordinate actions** to avoid conflicts and redundancy

Maintain consistency in data and decision-making

Handle errors gracefully without system-wide failures

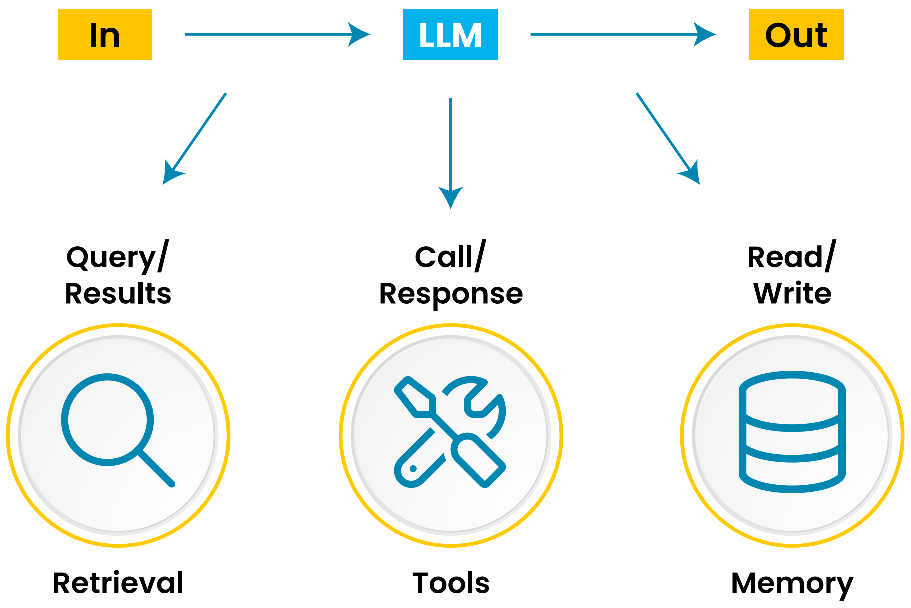
Scale operations as complexity increases

Think of protocols as the **"language"** and **"etiquette"** that agents use to **work together harmoniously.**

Today, AI agents trying to work together (whether LangChain, AutoGen, CrewAI, or custom solutions) speak their own dialect. This can of course cause chaos for developers. Standardized protocols have emerged that promise to fix some of this chaos: MCP (Anthropic), A2A (Google), ANP (Open Source Community), and ACP (IBM), LangChain Expression Language (LCEL). Each aims to solve the agent communication problem, but each approaches it from very different angles.

The augmented LLM

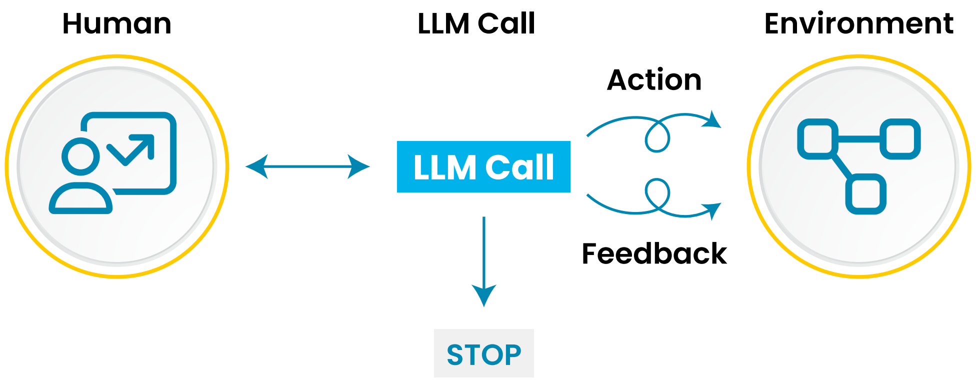
Agentic systems start with an LLM. But when you add tools, memory, and retrieval, they come alive. These models can search, decide, and adapt, all on their own.



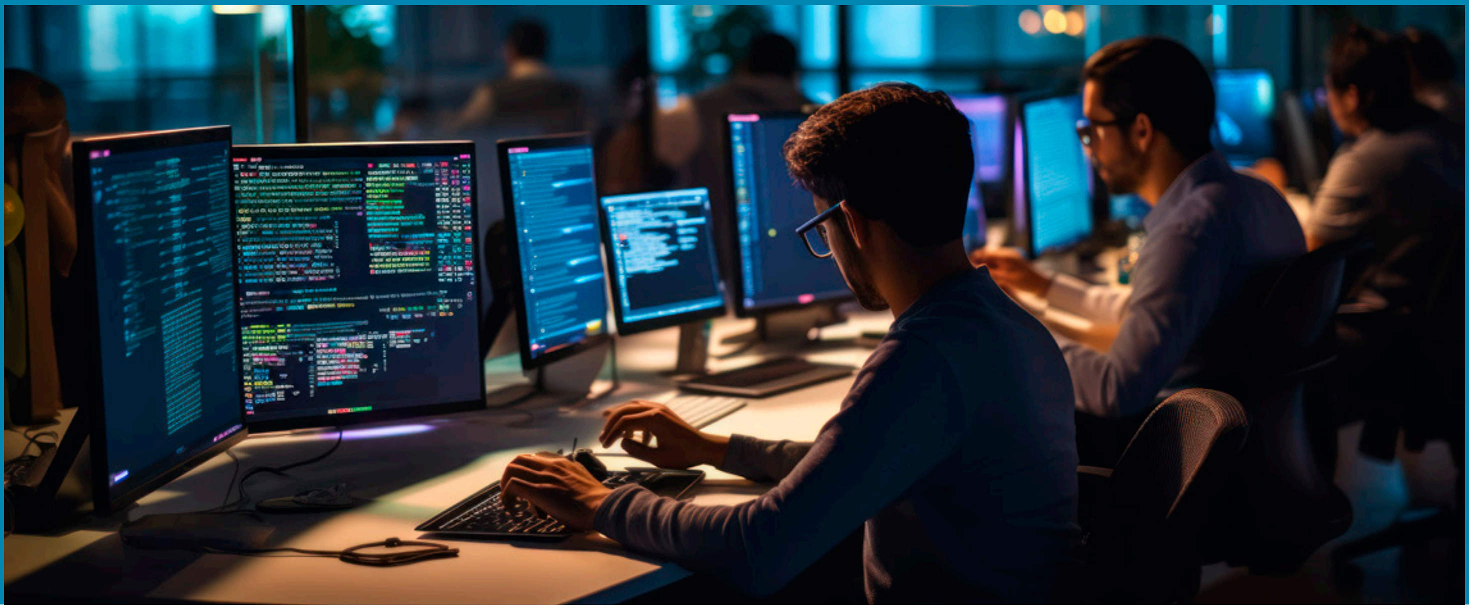
This visual was inspired by documentation from Anthropic and was created by the Black Kite Data Research team.

Autonomous agent

An autonomous agent begins with input from a human, either a direct command or a conversation. The agent then takes initiative: it plans, acts, observes the outcome, and adjusts accordingly. Throughout this loop, it interacts with tools and the environment, using real-time feedback to stay aligned with its goal. It can pause for human input when needed or stop based on predefined conditions. While the loop may seem simple, it enables dynamic, self-directed behavior.



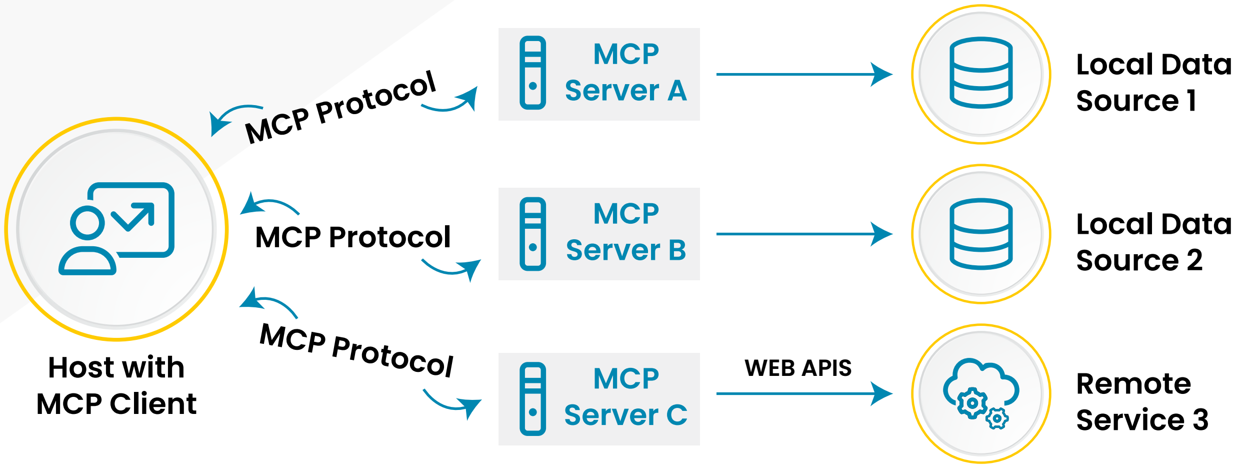
This visual was inspired by documentation from Anthropic and was created by the Black Kite Data Research team.



Model Context Protocol (MCP)

- MCP is the universal adapter that lets AI models plug into any data source. Released by Anthropic.
- MCP doesn't focus on agents talking to agents—instead, it standardizes how AI models connect to tools and data.
- MCP provides a universal, open standard for connecting AI systems with data sources, replacing fragmented integrations with a single protocol. It's like giving your AI assistant a master key to every database, API, and tool in your organization.

In short, we can say that it is the standardized version of connecting LLMs to external data sources.



This visual was inspired by documentation from Anthropic and was created by the Black Kite Data Research team.

How It Works

MCP works on a client - server handshake: the server advertises capabilities (tools, resources, prompts) through JSON-RPC, and the client forwards those to the LLM host. Your class is one such MCP Server that surfaces a database. **(The "server" concept here does not have to be a real HTTP service; it can also be a local subprocess).**

Code element	What MCP expects	Narrative for readers
self.name (e.g., "company_database")	A unique server ID the client can display in its capability list.	Think of name as the USB descriptor your agent shows when it's plugged in.
self.resources list	Static registry of data blobs (not functions). Each item will later map to a resource object in the MCP spec.	These are the tables you're willing to expose.
Async handle_resource_request()	Implements the Resources RPC; the client calls it when the LLM wants rows.	One coroutine per resource keeps the server non-blocking.
describe_capabilities()	Mandatory initialize() response: advertises resources (or tools) with JSON-Schema so the LLM can validate its own requests.	Without this, the model has no contract—schemas are your guardrails.

Use Case

The Challenge: When a Major Cloud Provider Gets Compromised

Imagine it's 3 AM, and news breaks that a major cloud infrastructure provider has suffered a data breach. Your TPRM team needs to immediately answer: "Which of our 500+ vendors are affected, what's their exposure level, and what's our business impact?"

Traditional Approach Problems:

- 6 different vendor databases with separate APIs
- Security ratings from 3 different providers (each with custom integration)
- Financial data from D&B, compliance records from audit systems
- Manual correlation taking 4-6 hours while business risk escalates

The MCP-Enabled Solution: Instant Vendor Impact Intelligence

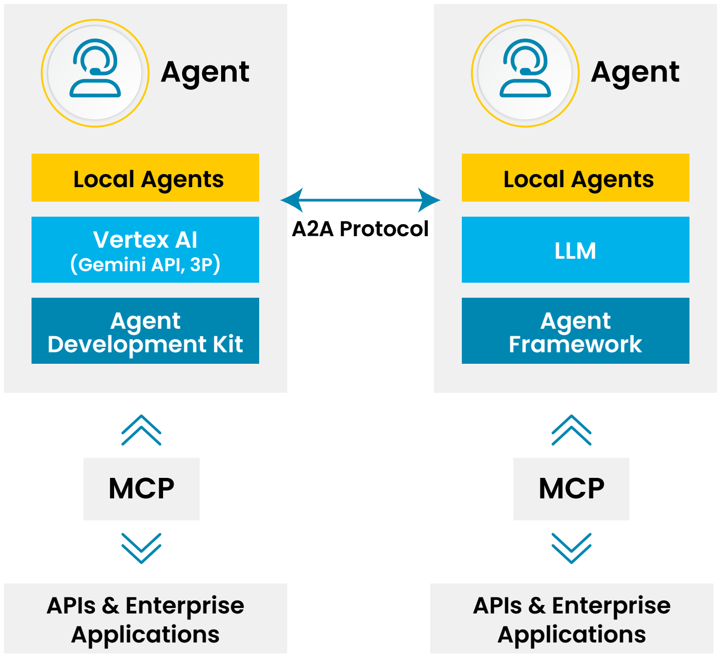
One simple query instead of 6 different API calls!

MCP is an ideal choice in this scenario because it requires standardized data access from multiple sources, not agent-to-agent coordination (A2A) or complex workflow orchestration (LCEL). When a breach occurs, you need instant data collection from vendor databases, security ratings, and compliance systems; MCP's “universal connector” approach combines 6+ custom API integrations into a single standardized interface that can pull real-time data from all sources simultaneously.

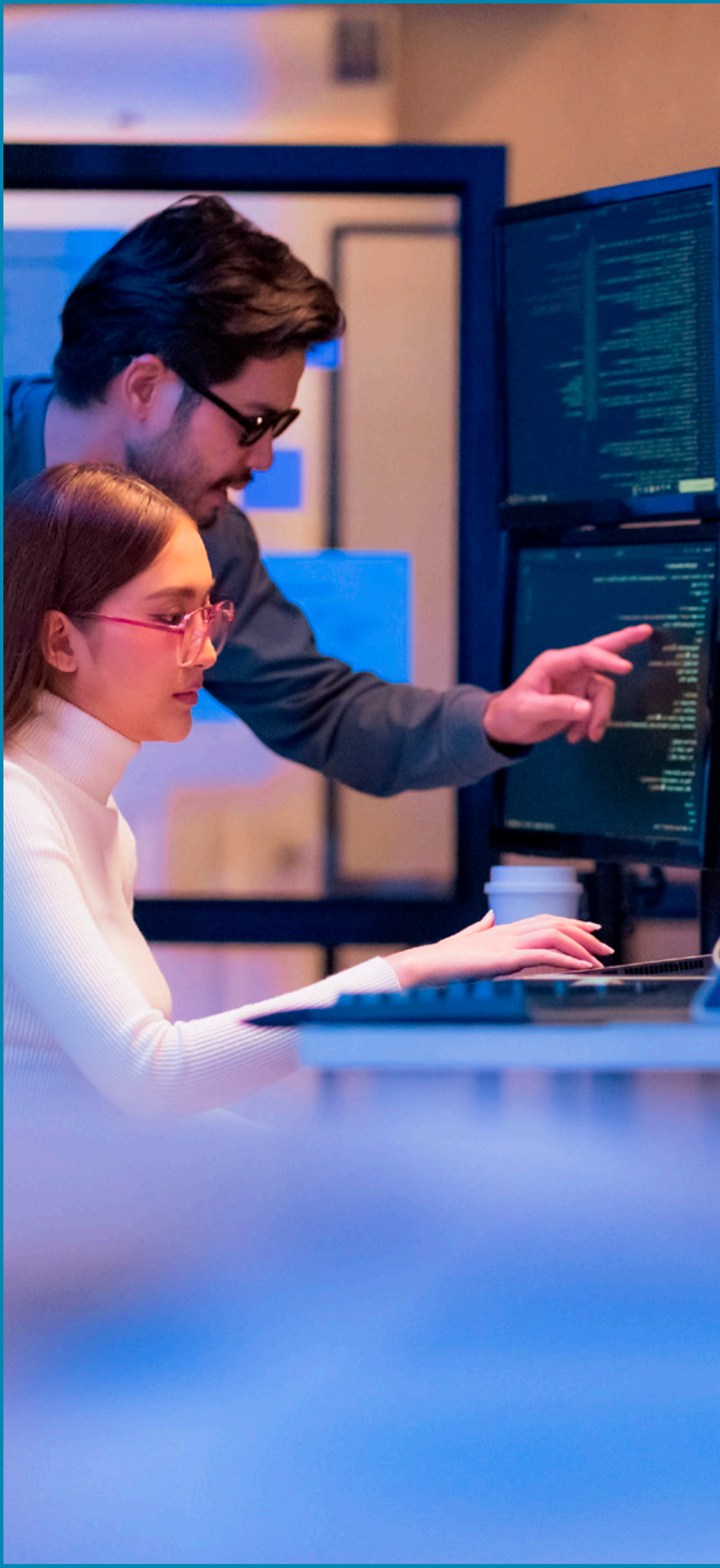
But of course, different protocols may be more useful in different scenarios, so let's examine other protocols!

Agent-to-Agent (A2A) Protocol

A2A enables AI agents to communicate with each other, securely exchange information, and coordinate actions across different platforms and frameworks. Unlike MCP which connects agents to data sources, A2A enables agents to collaborate as peers, sharing tasks and coordinating complex multi-step operations.



This visual was inspired by documentation from Google and was created by the Black Kite Data Research team.



How It Works

The Agent-to-Agent (A2A) protocol lets autonomous agents hand off tasks over plain HTTP: each message is a JSON-RPC envelope that carries a **Task,** optional streamed **Artifact,** and OAuth-scoped metadata, so one agent can discover another via its .well-known/agent.json card, send the task, and receive results in real time. In practice this means ***a threat-intel agent can trigger a vendor-risk agent just by posting a signed task object—no custom web-hooks or shared memory needed.***

Code element	What A2A expects	Narrative for readers
agent_card.json	Standardized capability advertisement via Agent Cards containing endpoints, authentication, and available services	Think of this as an agent's business card - it tells other agents "here's what I can do and how to reach me"
task_lifecycle	Structured task states: pending → in-progress → completed/failed with unique Task IDs	Like a project management system for agents - every collaboration has clear status tracking
secure_communication	JSON-RPC 2.0 over HTTPS with enterprise authentication (OAuth, mutual TLS)	Agents talk to each other like secure web services, not like informal chat
message_parts[]	Rich content exchange including text, files, artifacts with content-type negotiation	Agents can share anything from text responses to generated reports, images, or structured data
capability_discovery	Dynamic agent discovery and service registry for finding specialized agents	Like a phone book for agents - "I need a financial analysis expert" → finds and connects to appropriate agent

Use Case

The Challenge: Coordinated APT Campaign Detection

A sophisticated nation-state actor launches coordinated attacks across multiple organizations in the financial sector. Traditional threat hunting relies on each organization's isolated analysis, missing the broader campaign pattern.

The A2A-Enabled Solution:

- Network Behavior Agent (Bank A) detects anomalous traffic patterns
- Attribution Agent (Shared Intel Hub) receives IOCs for analysis
- Malware Analysis Agent (Bank B) analyzes related samples
- Campaign Correlation Agent synthesizes findings across all participants
- All participating agents receive coordinated threat intelligence

Why A2A specifically for this use case?

This scenario requires specialized agents to collaborate and coordinate their findings in real-time across organizational boundaries. MCP would only handle data access within each organization, and LCEL would orchestrate workflows within a single system, but only A2A enables secure, peer-to-peer agent collaboration where each agent maintains its expertise while contributing to collective defense.

Key Benefits:

- Cross-organizational intelligence sharing without exposing internal systems
- Specialized agents coordinate (network analysis + malware analysis + attribution)
- Real-time collaboration as threats evolve
- Maintains autonomy while enabling collective defense

LangChain Expression Language (LCEL)

LCEL is a declarative syntax for composing complex AI workflows that allows you to describe what you want to happen rather than how. It enables building sophisticated, multi-step analysis pipelines with built-in parallelization, streaming, and error handling for adaptive intelligence operations.

How It Works

LCEL is LangChain's declarative expression language that lets you describe an entire LLM workflow—prompts, tools, memory, error-handling—in a single, pipe-style line, which LangChain then compiles into an executable graph.

Code element	What LCEL expects	Narrative for readers
chain = step1 | step2 | step3	Declarative pipeline composition using pipe operator with Runnable interface	Think of this as assembly line instructions - you describe the sequence, LCEL handles the execution
RunnableParallel({...})	Parallel execution of independent analysis streams with automatic result aggregation	Like having multiple expert analysts work simultaneously on different aspects of the same problem
RunnableLambda(function)	Custom functions wrapped as Runnables with standardized input/output handling	Your domain-specific logic becomes part of the standardized workflow - no special integration needed
conditional_branch	Dynamic routing based on intermediate results using RunnableBranch	Workflows adapt like human experts - if confidence is low, route to additional validation
chain.with_fallbacks([...])	Built-in error handling and alternative execution paths for robust operations	When Plan A fails, Plans B and C automatically kick in - no manual intervention needed

This visual was inspired by documentation from LangChain and was created by the Black Kite Data Research team.

Use Case

The Challenge: Dynamic Threat Analysis Across Varying Confidence Levels

Your CTI team receives 1000+ threat indicators daily with varying quality, source reliability, and threat types. Each requires different analysis depth - high-confidence threats from trusted sources need fast processing, while ambiguous indicators need extensive validation.

The LCEL-Enabled Solution:

Suspicious Domain IOC arrives

- **LCEL Routing**

Medium confidence → Standard analysis path

- **LCEL Parallel Processing**

Context enrichment + Attribution analysis + Impact assessment

- **LCEL Aggregation**

High-risk APT29 infrastructure targeting finance sector

- **LCEL Automated Response**

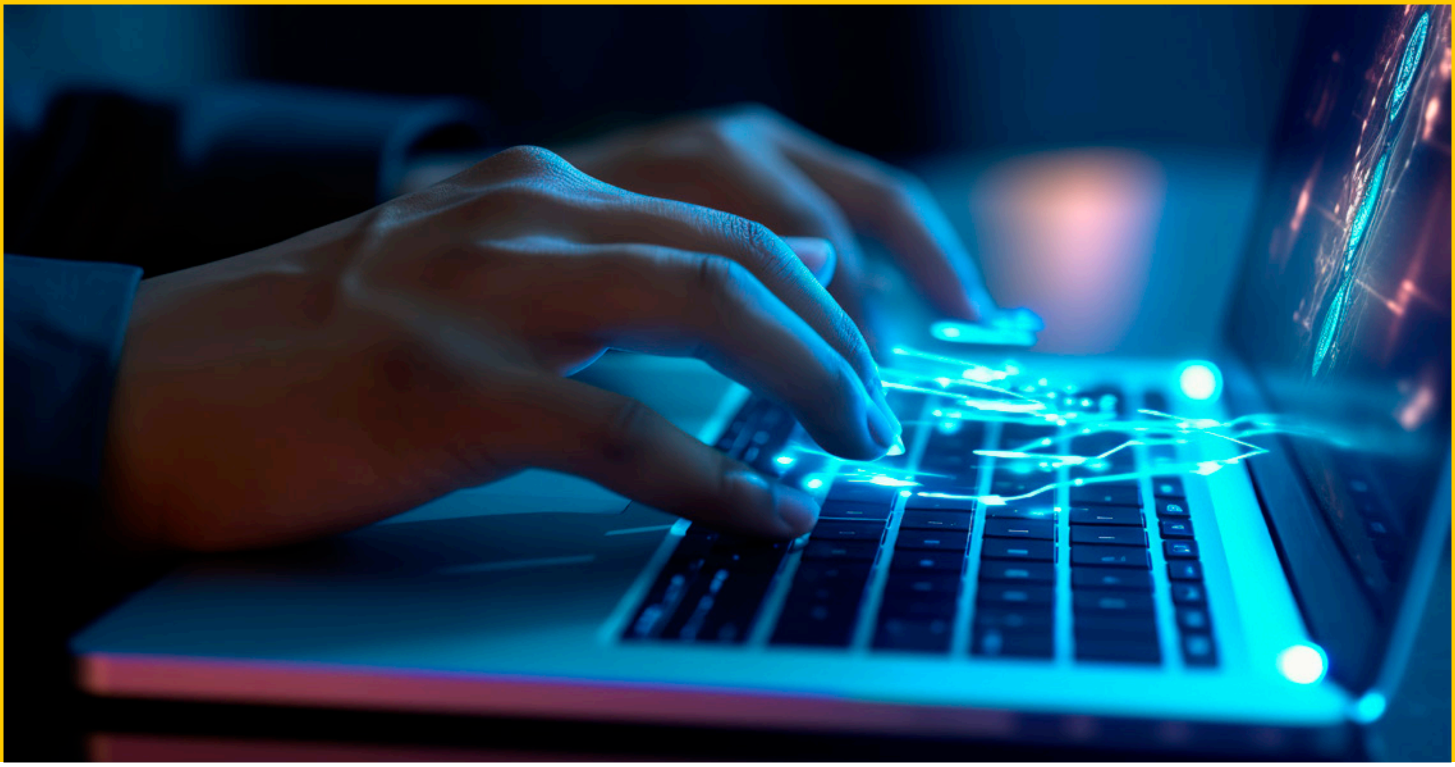
Immediate alert to SOC + automatic blocking rules deployed

Why LCEL specifically for this use case?

This scenario requires complex, adaptive workflows that change based on threat characteristics and analysis confidence levels. MCP would only provide data access to threat feeds, A2A would enable agent coordination, but only LCEL provides the declarative workflow orchestration needed to automatically route threats through different analysis pipelines based on confidence scores, threat types, and organizational priorities while maintaining parallel processing and fallback mechanisms.

Key Benefits:

- Adaptive routing based on threat confidence and type
- Parallel processing for faster analysis without workflow complexity
- Built-in fallbacks ensure no threat goes unanalyzed
- Declarative syntax makes complex workflows easy to modify as threats evolve
- Automatic optimization for streaming and performance



Conclusion:
Ready for AI Agents in TPRM?

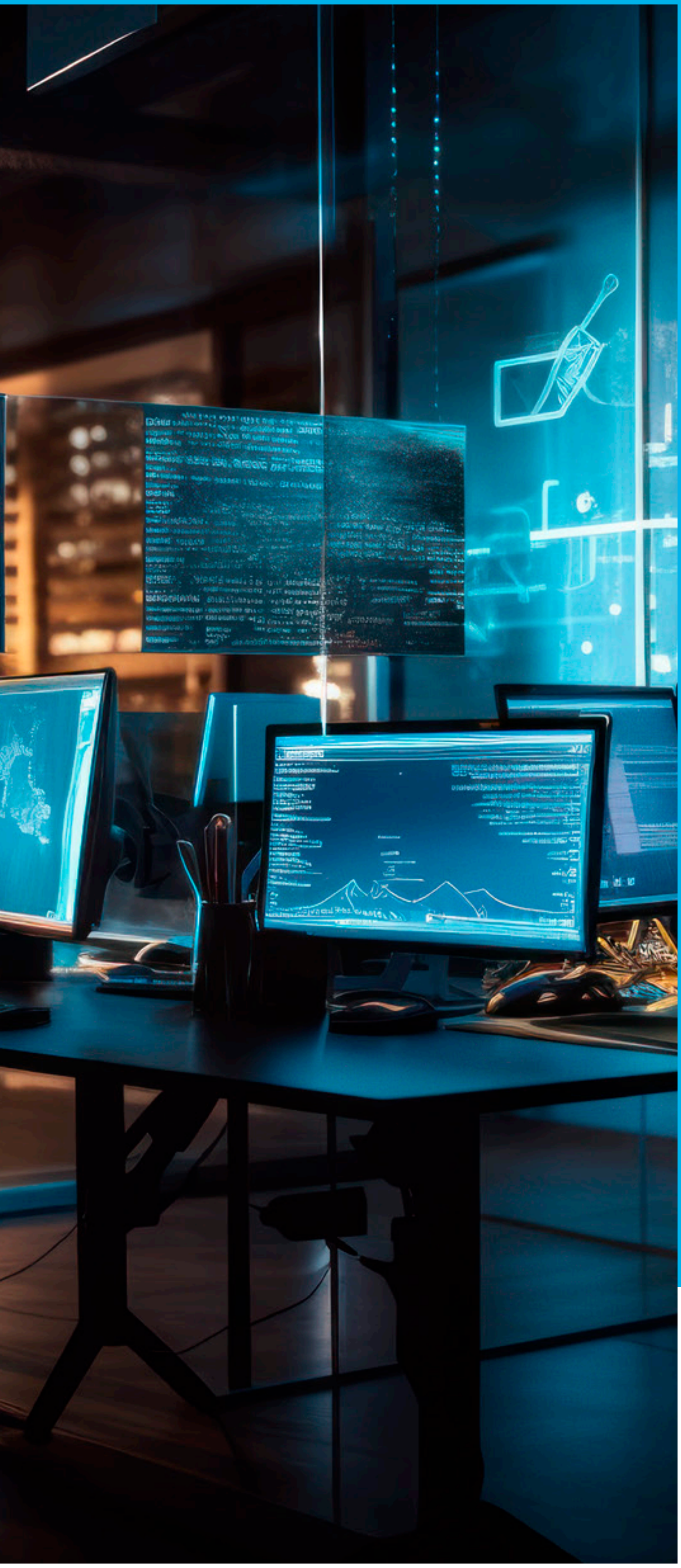
Let’s Wrap It Up!

We’ve covered a lot of ground, but before we call it a day, here’s a quick and snappy recap of how MCP, A2A, and LCEL stack up when it comes to multi-agent collaboration.

MCP, A2A, LCEL — Same Orchestra, Different Instruments:

- **MCP (Message Control Protocol)** - keeps the **wires clean and the traffic flowing.** Think of it as the traffic cop: stateless, simple, and focused on message routing and execution order.
- **A2A (Agent-to-Agent)** - is more like the **translator at the roundtable.** It ensures agents speak the same conceptual language, defining structured messages and intent formats.
- **LCEL (Language for Coordinated Emergent Logic)** - takes the role of **the composer,** enabling agents to orchestrate more complex reasoning and collaborative workflows in a readable, logic-first syntax.

These aren’t competing tools — they’re **different gears in the same engine,** each solving a layer of the coordination puzzle.



So… Are We There Yet?

While these protocols lay down the tracks, the train is still learning to run without bumping into itself.

Some key challenges remain:

- **Context isn't sticky yet:** Agents still lose track of long-term memory or shared understanding in dynamic tasks.
- **Tool use isn't fully plug-and-play:** Autonomy often depends on tightly tuned prompts or rigid APIs.
- **Graceful failure is rare:** Agents struggle with recovery and improvisation when things go sideways.
- **Collaboration lacks depth:** Reasoning about others' beliefs, knowledge, or goals — the so-called "theory of mind" — is still in its early days.
- **Security still has blind spots:** Agents can unintentionally leak sensitive data, misuse tools, or fall for adversarial prompts. With great autonomy comes great attack surface. Protocols help organize the system — but they don't guarantee it's secure.

These protocols take us several steps closer to seamless agent collaboration — but they don't solve the whole game. Coordination is easier, but not effortless. As we keep building better brains behind the agents, these frameworks will become even more powerful.

Want to learn more about protecting your business against today's fast-changing landscape of cyberthreats?

Schedule a consultation with Auxis' cybersecurity managed services experts today!

Or, **visit our resource center** for more cybersecurity tips, strategies and success stories.

BLACK KITE

auxis

A Grant Thornton US company