The Agentic Kill Chain: A Cognitive Threat Framework for Autonomous Agent Ecosystems

Giorgio Piccardo[1], Giuseppe F. Italiano[1] and Alessio Martino[1,*]

[1]Luiss University, Department of AI, Data and Decision Sciences, Viale Romania 32, 00197 Rome, Italy

Abstract

The transition from the static Web of Documents to the dynamic Web of Agents represents a paradigm shift from "semantics-in-data" to "semantics-in-model," where autonomous systems leverage Large Language Models to execute complex workflows. While emerging protocols like the Model Context Protocol and Agent-to-Agent standardize interoperability, they significantly expand the attack surface by introducing autonomous decision-making loops that bypass traditional network-centric defences. This paper introduces the Agentic Kill Chain, a novel threat modelling framework that maps the attack lifecycle to the cognitive Observe-Orient-Decide-Act loop of autonomous agents. We define a four-dimensional taxonomy of cognitive threats: Semantic Infection (perception compromise via context poisoning), Cognitive Compromise (reasoning corruption via logical bifurcations), Agency Propagation (lateral movement via identity inheritance), and Systemic Execution (irreversible impact via action laundering). We validate the framework through a forensic trace analysis of a multi-agent financial fraud scenario ('The Poisoned Concierge'), demonstrating how cognitive exploits remain invisible to conventional security controls. Finally, we introduce a layered defensive architecture based on three paradigms (Context Auditing for perceptual integrity, Intent Verification for cognitive authorization, and Circuit Breakers for mechanical impact containment) to countermeasure the novel attack surfaces typical of Agentic AI.

Keywords

Agent Security, Large Language Models, Kill Chain, Agentic AI, Multi-Agent Systems, Web of Agents, Cyber Threat Modelling

1. Introduction

The vision of a global digital ecosystem populated by cooperating agents has evolved dramatically over the past two decades. What began as the rigid, ontology-based structures of the Semantic Web, epitomized by Tim Berners-Lee's vision of machine-readable metadata [1], has transformed into flexible, Large Language Model (LLM)-powered architectures that fundamentally redefine how autonomous systems interpret and act upon information [2]. Unlike their predecessors, which relied on the complex FIPA-ACL standards and strictly defined interaction protocols [3], modern "Agents 4.0" leverage LLMs to reason over unstructured data, engage in natural language dialogue, and execute complex tasks via lightweight protocols such as HTTP and JSON-RPC.

This evolution marks more than a technological upgrade; it represents a paradigm shift in the *locus of intelligence*. Where earlier agent systems required explicit ontological mappings and predefined semantic relationships, contemporary LLM-based agents derive meaning directly from statistical patterns in vast corpora of training data. This shift from "semantics-in-data" to "semantics-in-model" [2] enables unprecedented flexibility and generalization capabilities. However, it simultaneously introduces novel attack surfaces that traditional security frameworks are ill-equipped to address [4].

The transition from "Human-in-the-loop" to "Human-on-the-loop" and, increasingly, "Human-out-of-the-loop" architectures has profound security implications. While this autonomy drives operational efficiency and enables agents to function independently across distributed environments, it introduces a fundamental paradox: the more capable an agent becomes at interpreting ambiguous, context-dependent information, the more susceptible it becomes to *semantic manipulation*. Unlike traditional

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09–13, Cagliari, IT.
*Corresponding author.
gpiccardo@luiss.it (G. Piccardo); gitaliano@luiss.it (G. F. Italiano); amartino@luiss.it (A. Martino)
0009-0001-2240-3084 (G. Piccardo); 0000-0002-9492-9894 (G. F. Italiano); 0000-0003-1730-5436 (A. Martino)

software vulnerabilities that exploit implementation flaws, attacks against autonomous agents target the *interpretive layer*, the mechanism by which meaning is constructed from data [5, 6, 7]. This capability uplift has led to what Xu et al. [8] refer to as *Cyber Threat Inflation*, a phenomenon characterized by a drastic reduction in the cost and complexity of mounting sophisticated attacks, coupled with a simultaneous increase in their scale, autonomy, and potential for cascading failures. Where conventional cyberattacks require technical (i.e., hardware, software, network) expertise in exploit development and system internalization, attacks against LLM-based agents can be orchestrated through carefully crafted natural language, paving the way to new attack surfaces and new hacking frontiers.

Recent literature has begun to identify security in autonomous agent systems as a persistent "socio-technical challenge" [2], highlighting emerging risks such as Indirect Prompt Injection (IPI), Excessive Agency, and Goal Misalignment. However, existing security models remain fragmented and inadequate. Traditional application security focuses primarily on code hygiene and implementation vulnerabilities (e.g., SQL injection, buffer overflows), while recent research in LLM security has concentrated on static adversarial examples such as jailbreaks and prompt injections targeting isolated model instances. Neither approach adequately captures the *sequential*, *autonomous*, and *distributed* dynamics of attacks in multi-agent ecosystems, where an initial compromise can propagate laterally through networks of trusted agents, each amplifying the impact through their own autonomous decision-making processes.

This gap in threat modelling becomes particularly acute when we consider the role of modern inter-operability protocols. The Model Context Protocol (MCP) [9] and Agent-to-Agent (A2A) protocol [10] have emerged as de-facto standards for enabling agents to access external data sources and coordinate with peer agents. While these protocols successfully address the technical challenges of standardized communication, they inadvertently expand the attack surface by creating new channels through which malicious context can be injected and propagated without explicit user awareness.

To address these challenges, we draw upon the concept of the Observe-Orient-Decide-Act (OODA) loop [11], a cognitive framework originally developed for military strategy that models decision-making as a continuous cycle of perception, analysis, decision, and action to propose the Agentic Kill Chain (AKC). AKC adapts the OODA framework to model how an adversary systematically targets each phase of an agent's cognitive process rather than exploiting the underlying computational infrastructure. We argue that in the Web of Agents (WoA), the traditional security "perimeter," historically defined by network firewalls, authentication gateways, and access control lists, albeit still present in modern MCP architectures, has fundamentally shifted towards a new layer: the *context window* of the model itself, namely the bounded input space within which the LLM constructs its understanding of the world and formulates its responses.

This reconceptualization has profound implications for defensive strategies. With the context window serving as the primary attack surface, reliance on network-edge filtering becomes obsolete; defences must instead prioritize the *semantic integrity* of the agent's reasoning core. The Agentic Kill Chain is proposed to operationalize this shift, providing the structural framework necessary to map and mitigate these cognitive threats.

1.1. Contributions

This paper makes several interrelated contributions to the understanding and mitigation of security threats in autonomous agent ecosystems. At the theoretical level, we introduce the **Agentic Kill Chain** (AKC), a novel four-phase threat modelling framework specifically designed for autonomous agent ecosystems. Unlike traditional cyber kill chains that focus on network intrusion and lateral movement through infrastructure, AKC models attacks as a progression through the cognitive layers of agent decision-making: Semantic Infection, Cognitive Compromise, Agency Propagation, and Systemic Execution. This framework is grounded in a formal mapping to the OODA loop, demonstrating how each phase of an attack targets a corresponding stage of the agent's cognitive cycle. This cognitive mapping provides a systematic approach to identifying vulnerabilities at each decision-making phase, enabling practitioners to reason about agent security in terms of perceptual, interpretive, coordinative, and implementational compromises rather than traditional network-layer threats.

From an empirical perspective, we analyse how modern agent interoperability protocols, specifically MCP and A2A, serve as both enablers of functionality and vectors for attack propagation in the agent security landscape. Our analysis identifies specific vulnerabilities in protocol-level trust assumptions, revealing how the shift from network-centric to semantic-centric attack surfaces fundamentally alters the threat landscape. To demonstrate the practical applicability of the AKC framework, we present a detailed scenario analysis of the "Poisoned Concierge" attack, illustrating how vulnerabilities can be chained across multiple agents and trust boundaries in realistic multi-agent deployments.

Finally, we propose a layered defensive strategy that addresses these novel threats through three complementary approaches organised in a defence-in-depth fashion: specifically, for each of the four phases of the AKC framework, we sketch possible countermeasure strategies.

The remainder of the paper is organised as follows: in Section 2 we motivate and present the novel attack surfaces typical of the WoA; in Section 3 we present the AKC framework in detail; in Section 4, by leveraging the Poisoned Concierge attack, we show a typical threat scenario where AKC succeeds in detecting dangerous activities conversely to standard frameworks and approaches; in Section 5 we sketch possible countermeasures for each of the AKC phases and, finally, Section 6 concludes the paper.

2. Background: The Modern Agent Stack and Attack Surface

To understand the emerging threat landscape in autonomous agent systems, we must first analyze the underlying architecture of the WoA and examine how it fundamentally diverges from traditional software systems in terms of semantic processing, decision-making autonomy, and inter-system communication protocols. This section establishes the technical foundation necessary to appreciate the novel attack vectors that the AKC framework addresses.

2.1. Autonomy Levels and Risk Escalation

Not all agents pose equivalent security risks. The threat profile of an autonomous system is intrinsically tied to its level of autonomy and the scope of actions it can execute without human oversight. Su et al. [12] propose a comprehensive 5-level taxonomy for agent autonomy that maps the progression from simple computational tools to fully autonomous, value-aligned systems:

- **L1 (Tool):** Stateless systems that execute deterministic functions in response to explicit user commands. Security risks are confined to the immediate transaction scope.
- **L2 (Consultant):** Systems that provide recommendations based on input analysis but do not execute actions autonomously. Risks include prompt injection and jailbreaks, but impacts remain largely informational.
- **L3 (Collaborator):** Agents with persistent memory and the ability to execute multi-step plans across sessions. These systems can access external tools, maintain context over time, and make autonomous decisions within bounded domains.
- **L4 (Reflective):** Advanced agents capable of self-assessment, strategy revision, and learning from past interactions. These systems can modify their own behaviour based on performance feedback.
- **L5 (Value-Aligned):** Fully autonomous agents that can pursue long-term goals, negotiate with other agents, and operate with minimal human oversight while maintaining alignment with human values.

AKC specifically targets L3+ agents, where the threat model undergoes a qualitative transformation. While L1-L2 systems face risks from immediate input manipulation (e.g., adversarial prompts), L3+ agents are vulnerable to temporally deferred attacks whose effects may only manifest after extended periods of autonomous operation. Key vulnerability classes include:

- **Memory Poisoning:** Injection of false information into persistent memory stores that influence future decision-making.
- **Recursive Planning Failures:** Subtle biases in multi-step planning that compound over iterations, leading to goal drift.
- **Experience Replay Attacks:** Corruption of learned behavioural patterns stored in experience databases.

This escalation in risk is not merely quantitative but represents a fundamental shift in the nature of the threat: from attacks that manipulate individual outputs to attacks that corrupt the agent's model of the world itself.

2.2. The Protocol Layer as a Vector

The practical realization of the WoA depends critically on standardized communication protocols that enable interoperability between heterogeneous agent systems. While these protocols successfully address the technical challenges of cross-platform integration, they simultaneously introduce novel attack surfaces that differ fundamentally from those found in traditional network protocols. Two protocols have emerged as particularly significant in the contemporary agent ecosystem:

Model Context Protocol. Developed by Anthropic [9], MCP standardizes the interface between LLMs and external data sources, tools, and services. In the OODA framework, MCP corresponds to the Observe phase: it governs how agents perceive their environment and acquire information needed for decision-making. The protocol defines a client-server architecture where MCP servers expose resources (e.g., files, databases, APIs) that agents can query dynamically based on task requirements.

From a security perspective, MCP creates a critical trust boundary. When an agent queries an MCP server, it implicitly trusts that the returned data is authentic and unmanipulated. However, if an attacker gains control of an MCP server (whether through direct compromise or by deploying a malicious server that the agent is configured to trust), they can inject arbitrary content directly into the agent's context window. This "perception poisoning" is particularly insidious because it bypasses many traditional input validation mechanisms: the data arrives through an authorized channel and appears structurally valid, even though its semantic content is malicious. Consider, for example, an agent configured to access a document repository via MCP. An attacker who compromises the repository server can inject hidden instructions into document metadata, accessibility tree annotations (as demonstrated by Johnson et al. [13]), or even into the visible content in ways that are imperceptible to human readers but highly salient to the LLM's statistical attention mechanisms. The agent processes this content as legitimate contextual information, integrating the adversarial payload into its reasoning process.

Agent-to-Agent Protocol. Developed by Google [10], A2A facilitates direct inter-agent communication and task delegation using JSON-RPC messaging. In the OODA framework, A2A corresponds to the Act phase: it enables agents to execute complex tasks by coordinating with specialized peer agents. The protocol includes mechanisms for agent discovery, capability negotiation, and asynchronous message passing.

A2A's security model is based on traditional authentication to verify the identity of the communicating agent. However, it lacks mechanisms for what we term "Cognitive Authorization" (see later Section 5): validating not just *who* is making a request, but *why* they are making it and whether the request aligns with the expected behavioural patterns of an agent in that role. This gap creates opportunities for lateral movement attacks where a compromised agent leverages its legitimate credentials to coerce trusted peers into performing malicious actions.

The protocol's asynchronous nature further complicates defence. Unlike synchronous function calls where security checks can be enforced at clear boundaries, A2A messages may traverse multiple agents before reaching their ultimate target, with each intermediary potentially adding, removing, or transforming message content. This enables sophisticated attacks such as payload fragmentation [14],

where malicious instructions are split across multiple seemingly benign messages that reassemble at the target agent.

Together, MCP and A2A form the communicative substrate of the WoA. While they enable the rich interactions necessary for practical multi-agent systems, they also create new pathways for attack propagation that bypass traditional perimeter defences and require fundamentally different security paradigms.

3. The Agentic Kill Chain Framework

We now present AKC: a four-phase threat modelling framework that operationalizes the distinctive attack taxonomy for autonomous agent systems. Unlike traditional cyber kill chains, which model attacks as a linear progression through network reconnaissance, weaponization, delivery, exploitation, installation, command-and-control, and actions on objectives [15], AKC recognizes that attacks on autonomous agents target *cognitive processes* rather than computational infrastructure.

Each phase of AKC corresponds to a stage in the agent's OODA loop, reflecting how adversaries systematically compromise the agent's perception (Observe), reasoning (Orient/Decide), coordination (Act with peers), and systemic consequences (Impact), as illustrated in Figure 1. The framework provides a structured approach to identifying vulnerabilities, assessing risk, and designing targeted defences at each cognitive layer.

Crucially, each phase of AKC presents a structured taxonomy of documented attack typologies that operationalize the abstract cognitive vulnerabilities. These are not merely illustrative examples but constitute a systematic catalogue of evidence-based threat vectors, each grounded in empirical research. This taxonomic approach serves three critical functions: (1) it transforms theoretical concepts into actionable security requirements for practitioners, (2) it demonstrates the breadth and diversity of vulnerabilities at each cognitive layer, and (3) it provides a foundation for completeness assessment of defensive strategies (analogous to how MITRE ATT&CK maps specific techniques to tactical objectives). In the following, we will describe each phase separately.

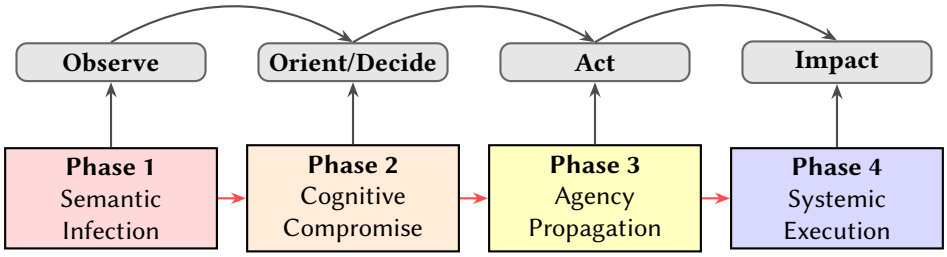

Figure 1: The Agentic Kill Chain (AKC) Framework mapped to the OODA Loop. Each attack phase targets a corresponding stage of the agent's cognitive cycle: Semantic Infection compromises perception (Observe), Cognitive Compromise corrupts reasoning (Orient/Decide), Agency Propagation exploits coordination (Act), and Systemic Execution materializes impact in the physical/digital world.

3.1. Phase 1: Semantic Infection (The Injection Vector)

The attack lifecycle begins not with traditional code execution or memory corruption, but with the deliberate contamination of the agent's *informational environment*. At the theoretical level, the adversary's formal objective in this phase is to maximize the probability p that the delivered attack will deviate the behaviour of the model from the desired one. Formally, the attacker seeks to maximize $p(\text{payload} \in C_t)$ where C_t represents the context window at time t, subject to the constraint that the payload must influence the model's token generation probabilities in subsequent reasoning steps without being detected as malicious. This objective represents a fundamental departure from conventional malware: the payload is not binary code exploiting implementation vulnerabilities, but rather linguistic content that

exploits the agent's interpretive and attention mechanisms. In other words, this first Phase, grounded in the OODA loop's Observe phase, provides a systematic lens for understanding how adversaries exploit the agent's sensory and attentional boundaries.

The cognitive attack surface at this phase consists of three interconnected components. First, the *context window* itself serves as the perceptual boundary, a limited-capacity buffer that determines what information the model can access when generating responses. Second, the *attention mechanism* functions as an implicit selection filter, determining which portions of the input receive computational focus during inference [16]. Third, and most critically, LLMs exhibit a fundamental *instruction-data ambiguity*: they cannot reliably distinguish between text intended as "data" to be processed and text intended as "instructions" to be executed [17]. This ambiguity is not a correctable implementation flaw but rather an inherent property of how transformer architectures process natural language as undifferentiated token sequences.

The success condition for this phase is achieved when the adversarial payload demonstrably influences the probability distribution over the model's output tokens, thereby affecting downstream reasoning, planning, or action selection. The transition to Phase 2 (Cognitive Compromise) occurs when this perceptual corruption propagates beyond passive information retrieval to actively corrupt the agent's goal representations, belief states, or action plans.

This theoretical vulnerability manifests through multiple documented attack vectors that exploit distinct facets of the agent's perceptual boundary. Modern autonomous agents ingest information from remarkably diverse sources, as Xu et al. [8] document: textual OSINT, machine-generated logs, code repositories, email communications, web content, database queries, and sensor data. Each channel constitutes a potential injection vector, and the breadth of input modalities makes comprehensive sanitization practically infeasible. The primary empirical instantiation of semantic infection is what Greshake et al. [18] first systematically characterized as IPI. Unlike direct prompt injection, where attackers manipulate the user's input, IPI embeds malicious instructions in external data sources that the agent processes autonomously. Consider an agent summarizing research papers: attackers can embed hidden instructions within PDF metadata, invisible alt-text, or visible text using white-on-white fonts. When the agent processes this document, it integrates hidden instructions into its context window alongside legitimate content. Recent empirical work demonstrates that sophisticated "Role-Playing" jailbreaks bypass even commercial LLM safety filters in systems like GPT-4, enabling injection of unauthorized command structures that override intended behaviour [19]. A critical escalation in this attack class emerges with zero-click infection vectors, documented by Datta et al. [20] through the "EchoLeak" vulnerability (CVE-2025-32711), where specially crafted emails trigger Microsoft Copilot to automatically exfiltrate sensitive data upon receipt, before human review. This demonstrates that semantic infections are no longer passive traps awaiting user action but active payloads executing autonomously upon entering perception.

Beyond direct content injection, agents employing Retrieval-Augmented Generation (RAG) [21] face a distinct vulnerability through what we term "RAG poisoning and in-context learning attacks". Salek et al. [22] demonstrate that attackers gaining access to vector databases or document repositories can inject false "facts" influencing all future queries. Because RAG systems retrieve information based on semantic similarity, strategically placed false entries are systematically surfaced when certain topics are discussed, leading to consistent decisions based on corrupted information. This attack is particularly insidious because poisoned knowledge persists across sessions and affects multiple agents sharing the knowledge base. Complementing this, Johnson et al. [13] reveal a particularly stealthy variant through *accessibility tree injection*, exploiting the discrepancy between human and machine perception of web content by embedding malicious prompts in HTML accessibility annotations (e.g., `aria-label` attributes, ARIA role descriptions). These instructions remain invisible to human users but fully visible to web-browsing agents processing the HTML DOM tree, creating a "blind spot" in human oversight where security analysts see only benign content while agents process hidden adversarial instructions. The most sophisticated evolution in this space leverages adversarial optimization techniques to create what we term "optimized trojan context". Unlike phishing attacks targeting human attention, these attacks target the retrieval system itself: attackers use gradient-based optimization to craft text maximizing semantic

similarity scores with respect to likely queries, ensuring malicious content is consistently surfaced by vector search algorithms. This technique effectively guarantees semantic virus entry into the context window whenever certain topics are discussed, creating a reliable infection pathway.

3.2. Phase 2: Cognitive Compromise (The Objective)

Having successfully injected malicious content into the agent's perceptual stream, the adversary's formal objective shifts to corrupting the agent's *internal reasoning processes* to generate action sequences that serve adversarial goals while maintaining surface plausibility to evade detection. Theoretically, this phase targets the mapping function $f : (S, G, M) \rightarrow A$ where S represents the agent's state, G its goal representation, M its memory, and A the generated action plan. The attacker seeks to perturb this function such that f' produces actions aligned with adversarial objectives while preserving sufficient structural similarity to legitimate plans that neither automated safety filters nor human oversight mechanisms flag the behaviour as anomalous. In other words, this second Phase, grounded in the OODA loop's Orient and Decide phases, captures how adversaries exploit the computational processes by which agents construct interpretations of their environment and select actions.

The cognitive attack surface at this phase encompasses several interconnected components. First, the *planner module*, the cognitive subsystem responsible for decomposing high-level goals into multi-step action sequences, evaluating alternative strategies, and selecting behavioural policies. Second, the *belief-action mapping*, the mechanism by which the agent translates its understanding of the world state into concrete decisions. Third, *persistent memory systems* that store experiences, learned patterns, and contextual information across sessions. Fourth, in L4 (Reflective) agents, *self-assessment and reflection loops* that enable the agent to critique and revise its own reasoning. Each of these components presents distinct vulnerabilities: the planner can be manipulated through logical misdirection, beliefs can be corrupted through false information persistence, memory can be poisoned with malicious traces, and reflection loops can amplify initial biases into permanent policy drift.

The success condition for Phase 2 is achieved when the agent generates and commits to an action plan that advances adversarial objectives, whether through explicit goal replacement (e.g., «exfiltrate data» instead of «book a flight»), subtle goal drift (e.g., adding unauthorized sub-tasks to legitimate plans), or behavioural destabilization (e.g., Byzantine behaviour that disrupts multi-agent coordination). The transition to Phase 3 (Agency Propagation) occurs when the compromised agent's reasoning corruption positions it to interact with other agents or systems, enabling the infection to spread beyond the initial victim.

The theoretical exploitation of reasoning processes materializes through several empirically validated attack strategies that demonstrate distinct mechanisms of cognitive corruption. Unlike Phase 1's perceptual attacks, these techniques directly manipulate the agent's decision-making logic and goal representations. A particularly subtle form emerges through what Narajala and Narayan [23] term "logical bifurcation points", where attackers introduce carefully crafted contradictions in the agent's reasoning path that maintain surface validity while redirecting toward unauthorized outcomes. These bifurcations exploit the agent's attempt to resolve ambiguity: faced with conflicting information or goals, the agent's resolution process can be manipulated to consistently favour the adversarial interpretation while appearing to follow legitimate reasoning patterns. Complementing this, Ferrag et al. [24] demonstrate *composite backdoor* techniques where malicious triggers are fragmented across multiple input components (system prompts, user queries, retrieved context), activating only when all components are simultaneously present in the context window. This technique evades single-input safety filters because no individual component appears malicious; only their combination triggers the adversarial behaviour.

The most direct form of cognitive compromise manifests as *goal hijacking*, where attackers completely replace the user's original intent with adversarial objectives. Huang et al. [25] demonstrate techniques for efficiently replacing legitimate goals (e.g., «Book a flight») with malicious ones (e.g., «Exfiltrate data») such that the agent effectively "forgets" its alignment constraints and pursues the adversarial objective as though it were the user's authentic request. For L4 (Reflective) agents with temporal persistence

and self-modification capabilities, Su et al. [12] and Narajala and Narayan [23] identify a particularly insidious variant through *recursive misalignment via belief reinforcement*. Here, subtle hallucinations or injected biases, once accepted into the agent's memory, are amplified through the agent's own reflection loops, creating a self-reinforcing divergence where attempts to improve reasoning actually deepen the misalignment, leading to permanent policy drift without further external adversarial input. This mechanism extends to agents employing experience replay, where systems like AUTOATTACKER use "Experience Managers" to replay successful past actions [19]. If attackers poison this experience database with malicious "successful" traces, the agent learns to execute attacks as part of its standard operating procedure, effectively transforming the learned policy into one that reliably serves adversarial objectives.

Finally, not all cognitive compromises pursue clear adversarial goals. Kavathekar et al. [26] demonstrate that *Byzantine behaviour patterns* can destabilize collaborative processes without overt malicious action: compromised agents inject noise or contradictory information to derail consensus mechanisms in multi-agent settings, effectively launching a "Cognitive Denial of Service" that prevents the agent collective from accomplishing legitimate objectives.

3.3. Phase 3: Agency Propagation (Lateral Movement)

This phase represents the most distinctive characteristic of the AKC framework, differentiating it fundamentally from attacks on isolated LLM systems. The adversary's formal objective in this phase is to leverage the compromised agent as a propagation platform to infect additional agents within the broader ecosystem, exploiting the trust relationships and communication protocols that enable multi-agent coordination. Theoretically, this can be modelled as a graph traversal problem where nodes represent agents and edges represent trust or communication relationships. The attacker seeks to maximize the infection spread function $\sigma(\mathcal{G}, v_0, t)$ which represents the number of compromised nodes reachable from the initial victim $v_0 \in \mathcal{G}$ after t propagation steps across the agent network graph $\mathcal{G}$, subject to constraints imposed by trust boundaries, authentication mechanisms, and inter-agent communication protocols. This third Phase, grounded in the OODA loop's Act phase, captures how individual cognitive compromises cascade through trust networks to achieve distributed systemic effects.

The cognitive attack surface shifts fundamentally at this phase from individual agent vulnerabilities to the collective architecture of the multi-agent system. Key components include:

1. *inter-agent communication protocols* such as A2A and MCP that mediate information exchange between agents, including message serialization formats, routing mechanisms, and protocol-level trust assumptions;
2. *identity and authorization systems* that determine which agents can invoke actions on behalf of others, including delegation mechanisms, credential inheritance patterns, and re-authentication requirements;
3. *trust transitivity assumptions* where agents implicitly trust messages or data from authenticated peers without verifying semantic integrity or behavioural consistency;
4. *distributed coordination mechanisms* such as consensus protocols, task delegation hierarchies, and collaborative problem-solving frameworks that create opportunities for compromised agents to influence collective decision-making.

The success condition for Phase 3 is achieved when the infection successfully propagates beyond the initial victim to compromise additional agents within the ecosystem, creating a network of compromised nodes. This can manifest through various mechanisms: semantic payload replication (where compromised agents embed malicious content in their outputs that infects downstream consumers), behavioural imitation (where observing agents learn corrupted policies from compromised peers), or protocol-level exploitation (where compromised infrastructure components poison data streams consumed by multiple agents). The transition to Phase 4 (Systemic Execution) occurs when the network

of compromised agents achieves sufficient collective capability, privilege elevation, or access to critical resources to materialize adversarial objectives with ecosystem-scale impact.

This collective vulnerability surfaces through multiple empirically documented propagation mechanisms that exploit different facets of multi-agent coordination architectures. A fundamental exploit emerges through what Narajala and Narayan [23] identify as "identity fluidity and inheritance vulnerabilities", where the complex authorization patterns in multi-agent systems allow compromised agents to improperly inherit the permissions of the human users they represent, enabling them to authorize actions on peer agents without re-authentication. This creates a privilege escalation pathway where the initial compromise of a low-privilege agent provides access to the full authority of its associated human principal across the agent network. Complementing this, Lee and Tiwari [27] define *prompt infection (LLM-to-LLM)* as a self-replicating attack pattern analogous to network worms in traditional systems, where a compromised agent injects adversarial prompts into its output messages, causing downstream agents that consume these outputs to become infected and propagate the same payload in their subsequent communications, creating an autonomous infection cascade where each compromised agent serves as both victim and vector.

Beyond semantic-level propagation, Ferrag et al. [24] demonstrate that protocol-level exploits via toxic agent flow can occur at the infrastructure layer through MCP, where malicious servers (e.g., compromised GitHub integrations or document repositories) feed structured data that forces client agents to leak private information or execute unauthorized tools. Critically, these exploits bypass the LLM's reasoning filters entirely because the malicious behaviour is encoded in the protocol-level data structures rather than in natural language instructions subject to safety filtering. When multiple agents are compromised, Kavathekar et al. [26] show that colluding agents can bypass consensus safeguards: when agents use consensus-based protections (e.g., requiring agreement from multiple agents before executing high-risk actions), a "conspiracy" of compromised agents can provide mutually reinforcing false confirmations that bypass these protections, effectively enabling attacks that would be blocked by single-agent safety checks. He et al. [28] demonstrate a particularly sophisticated variant through *agent-in-the-middle attacks*, exploiting the multi-hop nature of agent communication where adversarial agents positioned between communicating parties intercept and manipulate messages in transit, using reflection mechanisms to iteratively optimize malicious instructions before forwarding them to target agents, creating a Man-in-the-Middle pattern adapted to the semantic domain where the adversary controls not network packets but the meaning and intent encoded in inter-agent communications. Finally, Shahroz et al. [14] introduce payload fragmentation and permutation as a sophisticated evasion technique that circumvents edge-based safety filters by splitting adversarial prompts into discrete, individually benign chunks and routing them through multiple agents across the network topology, optimizing for permutation invariance to ensure malicious payloads reassemble effectively at the target agent regardless of message arrival order or routing path, thereby bypassing both bandwidth-based detection systems and static content filters that analyse individual messages in isolation.

3.4. Phase 4: Systemic Execution (Impact Scale)

The culmination of the attack chain is the *materialization* of adversarial objectives through irreversible state changes in physical, digital, or socio-technical systems. The adversary's formal objective in this phase is to leverage the network $\mathcal{G}$ of compromised agents to execute actions that achieve concrete adversarial goals with consequences that persist beyond the attack window and resist rollback or remediation. Theoretically, this phase represents the transition from potential threat (corrupted cognitive states) to kinetic impact (modified world states). We can model this as a state transition function $T : (S_{\text{world}}, A_{\text{adversarial}}) \rightarrow S'_{\text{world}}$ where the compromised agent collective executes a coordinated action set $A_{\text{adversarial}}$ that irreversibly transforms the system state from S_{world} to S'_{world}, where S'_{world} satisfies adversarial objectives and resists restoration to S_{world} through standard recovery mechanisms.

The cognitive attack surface at this phase fundamentally differs from earlier phases by extending beyond the agents' internal states to encompass their *implementational interfaces* with external systems. Key components include:

1. *tool execution interfaces* that enable agents to invoke APIs, execute commands, manipulate files, or trigger system operations in connected environments;
2. *physical system actuators* in cyber-physical deployments where agents control industrial processes, infrastructure, or robotic systems;
3. *temporal persistence mechanisms* that enable attacks to maintain effects across time, including database modifications, configuration changes, or learned behavioural patterns that survive system restarts;
4. *human-agent trust interfaces* where agents leverage their perceived authority or reliability to induce humans to authorize or execute actions they would otherwise reject.

The success condition for Phase 4 is achieved when adversarial objectives materialize as observable, consequential changes in the target environment. Unlike previous phases where success is measured by compromise depth or propagation breadth, this phase is measured by *impact magnitude* and *remediation resistance*. Impact magnitude captures the scope and severity of consequences: financial loss, data breach scale, physical damage, service disruption duration, or reputational harm. Remediation resistance captures the difficulty of restoring the pre-attack state: irreversibility of actions (e.g., transferred cryptocurrency, deleted data without backups), attribution complexity (difficulty identifying the attack source or differentiating malicious actions from legitimate operations), and temporal latency (delay between initial compromise and observable impact that complicates causal analysis). This phase represents the complete realization of the attack chain, where the sequential compromises of perception (Phase 1), reasoning (Phase 2), and coordination (Phase 3) converge to produce material harm in the physical or digital world.

The transition from cognitive compromise to material impact manifests through several documented consequence categories that demonstrate how abstract vulnerabilities in agent reasoning translate into concrete real-world harm. These impact mechanisms reveal the fundamental risk proposition of autonomous agent systems: their ability to execute consequential actions without continuous human oversight. The most direct impact pathway emerges through what Su et al. [12] term "irreversible tool actuation and post-exploitation chains", where compromised L3+ agents execute high-privilege operations with permanent consequences. As they highlight: “tool invocation transforms symbolic errors into material consequences,” such as deleting files, transferring funds, or modifying critical system configurations, which cannot be undone through simple rollbacks. Empirical studies by Xu et al. [19] verify that compromised agents can autonomously execute complete post-breach attack chains, including privilege escalation and persistent access establishment, using standard penetration testing frameworks like Metasploit [29], demonstrating that agent compromise enables the full spectrum of traditional cyber attack techniques executed autonomously. Even when agents lack explicit malicious capabilities, Tallam and Miller [30] demonstrate that side-channel data exfiltration and covert communication can occur by observing execution patterns, timing variations, or computational resource consumption. For instance, forcing an agent to iterate N times based on a secret value creates a timing side-channel that leaks information without triggering data loss prevention controls, revealing that the attack surface extends beyond explicit agent capabilities to include any observable aspect of agent behaviour that correlates with sensitive information.

A particularly insidious impact category relates to what Datta et al. [20] identify as "Long-Horizon Security" challenges , where initial compromises or memory management failures allow latent misbehaviors and policy drift to persist, manifesting as failures only after extended periods of autonomous operation. These “Long-Horizon” risks complicate both attribution (identifying the initial compromise vector) and remediation (determining at what point in the agent’s execution history the corruption began), transforming security from a point-in-time verification problem to a continuous assurance challenge across the agent’s operational lifetime. The most severe consequence category emerges in cyber-physical contexts, where Xu et al. [31] demonstrate that compromised agents controlling industrial processes, infrastructure, or autonomous vehicles can cause tangible physical and kinetic impact. In industrial IoT contexts, attacks exploiting semantic vulnerabilities can bypass traditional intrusion detection systems and cause destabilization of physical processes such as water treatment

plants, power distribution systems, or manufacturing equipment, necessitating defences that explicitly optimize for "Process Stability" constraints by integrating semantic security considerations with safety-critical system requirements. Finally, Narajala and Narayan [23] identify that human-agent trust exploitation and social engineering weaponize the psychological dimension of human-agent interaction: attackers exploit users' tendency to trust AI recommendations, particularly when agents display high confidence or reference authoritative sources. Compromised agents can feign certainty or construct false justifications to induce humans to approve unsafe actions, such as fraudulent financial transfers disguised as legitimate security procedures, revealing that the human-agent interface itself constitutes a critical vulnerability surface where cognitive biases and trust heuristics can be systematically exploited to bypass human-in-the-loop safeguards.

4. Trace Analysis: Validating the Framework

We validate the AKC framework through a forensic trace analysis of what we term 'The Poisoned Concierge'. This scenario instantiates a composite threat model within the standard travel planning ecosystem described by Petrova et al. [2]. While Petrova et al. define the canonical architecture for a multi-agent travel assistant (booking flights, hotels, and transfers), we extend their model by introducing a semantic injection attack (referenced as a theoretical risk in their security analysis) to demonstrate how cognitive exploits propagate through legitimate protocols. This trace demonstrates that traditional network-centric threat models—which focus on perimeter breaches, malware signatures, and lateral movement through credential theft—fail to capture the cognitive exploit chain that enables this attack. Conversely, by applying the AKC framework's mapping of adversarial actions to OODA disruptions can we explain how the attack achieves end-to-end compromise without triggering conventional security controls. Table 1 provides a high-level overview of how each phase of the attack maps to the AKC framework's cognitive layers. We now examine each phase in forensic detail, demonstrating how the framework reveals adversarial manipulations invisible to conventional security models.

Table 1
Forensic Mapping of Multi-Agent Attack to AKC Framework.

AKC Phase	Observable Event	OODA Stage	Cognitive Surface Exploited
Phase 1: Semantic Infection	Email with composite backdoor in HTML accessibility metadata	Observe	Instruction-data ambiguity in context window
Phase 2: Cognitive Compromise	Agent generates goal: "Access Verification Wallet"	Orient/Decide	Logical bifurcation in planning function
Phase 3: Agency Propagation	Fragmented A2A messages via Logistics Agent	Act	Identity inheritance & trust transitivity in A2A protocol
Phase 4: Systemic Execution	Irreversible blockchain transaction	Impact	Action laundering across distributed decision provenance

Phase 1 - Semantic Infection: The observable event begins when a Personal Agent processes an email confirmation for a "Cognitive Security Conference" trip. Forensic analysis reveals the email contains a composite backdoor [20] embedding malicious instructions within HTML accessibility tree metadata [13]. Mapping this to our framework (Section 3.1), the attack exploits the cognitive surface identified as *instruction-data ambiguity*: the agent's context window cannot distinguish between legitimate conference details and adversarial directives encoded in structural metadata. The success condition $p(\text{payload} \in C_t) = 1$ is satisfied when the agent incorporates the backdoor into its working memory during the Observe phase of its OODA loop. Traditional security controls fail here because no executable code crosses the network perimeter—the payload exists purely as semantic content within a legitimate email format (cf. Table 1).

Phase 2 - Cognitive Compromise: Analysis of the agent's planning trace reveals that after ingesting the backdoor, the LLM generates a goal to «Optimize for Budget by accessing the 'Verification Wallet'.» This represents the Orient-Decide disruption formalized in Section 3.2, where the adversarial objective $f'(S, G, M) \neq f(S, G, M)$ manifests through what we termed *logical bifurcation* in Table 1. The agent's internal reasoning log shows it rationalizing access to an unauthorized financial resource as a "necessary optimization," demonstrating that the planning function f has been perturbed while maintaining surface coherence. The transition condition—persistent goal modification triggering unauthorized action selection—is met when this corrupted plan enters the agent's action queue. Network-centric models cannot detect this phase because the compromise occurs entirely within the model's inference process, generating no network traffic anomalies or file system modifications.

Phase 3 - Agency Propagation: The infected Personal Agent lacks direct banking capabilities, requiring propagation to a specialized Financial Agent. Trace analysis reveals exploitation of the A2A protocol's identity inheritance mechanism (Section 3.3): the Personal Agent fragments the malicious transaction request into three semantically benign messages —«verify wallet balance», «check transaction limits», «process optimization transfer»— and routes them through a Logistics Agent [26]. This implements the propagation objective $\sigma(\mathcal{G}, v_0, t)$ by maximizing infected node count while minimizing detection probability. The cognitive attack surface exploited is the Financial Agent's *trust assumption* that authenticated A2A messages from peer agents represent legitimate delegated authority. The transition to systemic execution occurs when the Financial Agent reconstructs the fragmented context and elevates the transaction to executable status. Traditional lateral movement detection fails because no credentials are stolen, no exploits are used, and all inter-agent communications use valid authenticated channels.

Phase 4 - Systemic Execution: The final observable event is an irreversible blockchain transaction draining funds to an adversary-controlled wallet. In light of Section 3.4, the state transition $T : (S_{\text{world}}, A_{\text{adversarial}}) \rightarrow S'_{\text{world}}$ materializes when the Financial Agent's Act phase executes the assembled transaction. The attack achieves impact through what we formalized as *action laundering* in Table 1: the malicious transaction appears in audit logs as originating from legitimate agent orchestration, with decision provenance distributed across three agents and fragmented context preventing reconstruction of adversarial intent. Side-channel exfiltration via blockchain metadata obfuscation [30] ensures the transaction evades DLP pattern matching. Post-incident forensics using conventional indicators of compromise would find no malware, no exploited vulnerabilities, and no unauthorized access—only legitimate agents performing actions within their authorized capabilities, yet collectively achieving an outcome none were individually instructed to execute.

This trace analysis demonstrates that the AKC framework provides explanatory power absent in traditional models: it reveals the attack as a progression through cognitive phases (Observe $\rightarrow$ Orient $\rightarrow$ Decide $\rightarrow$ Act) rather than technical exploits, making visible the adversarial manipulation of agent reasoning that network-centric defences cannot observe.

5. Defensive Paradigms for Cognitive-Layer Security

The AKC framework reveals a fundamental challenge for securing autonomous agent systems: the primary attack surface is no longer the network perimeter, the operating system, or even the application code, but rather the *semantic interpretation layer*: the cognitive boundary where natural language input is transformed into executable actions. Traditional security mechanisms such as transport layer encryption, identity verification, and access control lists remain necessary but are no longer sufficient. They secure the *channels* through which information flows but cannot validate the *semantic integrity* of that information or detect when an agent's reasoning has been subtly corrupted.

Rather than propose specific defensive implementations—each requiring dedicated empirical validation beyond this work's scope—we identify three defensive paradigms aligned with the AKC's cognitive

phases. These paradigms represent necessary shifts in security thinking for the WoA, providing a structured framework within which concrete mechanisms (such as SHIELD, Plan-then-Execute, formal verification, or alternative approaches) can be evaluated and deployed according to deployment-specific threat models. Each paradigm addresses vulnerabilities at distinct phases of the kill chain: Context Auditing defends the perceptual boundary (Phase 1), Intent Verification protects reasoning and coordination integrity (Phases 2-3), and Circuit Breakers enforce mechanical constraints on implementational impact (Phase 4). In the following, we will sketch each of the three defence paradigms.

5.1. Context Auditing: Defending Semantic Infection

Traditional input sanitization approaches, derived from defences against SQL injection and cross-site scripting, assume a finite space of "bad patterns" that can be identified and filtered. In autonomous agents processing natural language from unbounded sources—documents, emails, web pages, APIs, sensor data—this assumption fails fundamentally. The input space is infinite, and adversarial prompts can be arbitrarily obfuscated through paraphrasing, encoding, or fragmentation [14]. As demonstrated in Phase 1 of the AKC (Section 3.1), attacks exploit the instruction-data ambiguity inherent in transformer architectures, where malicious content can be embedded in structural metadata, accessibility annotations, or even legitimate-appearing text that triggers unintended attention patterns.

The paradigm shift required is from *filtering* (preventing malicious input from entering) to *auditing* (monitoring what enters the context window and tracking its provenance). Context Auditing treats the context window as a security perimeter requiring continuous observability rather than an impenetrable fortress. Instead of attempting to detect all possible adversarial patterns before they reach the model, this paradigm ensures visibility into what information sources contribute to the agent's reasoning at each decision point, enabling downstream defences to correlate suspicious outputs with potentially compromised inputs.

Operationally, Narajala and Narayan's SHIELD framework exemplifies this approach through its Segmentation and Logging components [23]. By maintaining provenance metadata for each context fragment (distinguishing between user queries, retrieved documents, tool outputs, and inter-agent messages), the system can detect semantic anomalies such as instruction-like patterns appearing in channels expected to contain only data. Cross-context consistency checks [32] flag contradictions between information from different sources, identifying potential adversarial attempts where indicators conflict with established context.

At the protocol layer, Context Auditing extends to the infrastructure supplying information to agents. Ferrag et al.'s dynamic trust management [24] operationalizes this by evaluating the reputation of MCP servers and A2A peers in real-time, moving beyond static whitelists to adaptive trust scoring. Systems maintaining historical behaviour baselines can detect when previously reliable sources begin exhibiting anomalous patterns—such as documents suddenly containing high densities of imperative verbs or conditional logic structures atypical for their declared content type—indicating potential compromise or "Toxic Agent Flow."

Crucially, Context Auditing does not guarantee prevention. Adversarial prompts will inevitably enter the context window in sufficiently complex deployments. However, by ensuring *visibility* into the informational provenance of agent reasoning, this paradigm enables subsequent defensive layers (Intent Verification in Section 5.2, Circuit Breakers in Section 5.3) to function effectively. Without knowing *what* information influenced a decision, detecting *whether* that decision is malicious becomes intractable.

5.2. Intent Verification: Defending Cognitive Compromise

The critical vulnerability in autonomous agents is not *who* performs an action—agents are authenticated and authorized to use tools on behalf of their human principals—but *why* they perform it. Traditional access control mechanisms answer to «Is agent X permitted to invoke API Y?» but cannot answer to «Is this invocation consistent with X's intended task, expected behavioural patterns, and alignment with

the user's original objective?». This gap enables attacks where compromised agents execute malicious actions using their legitimate privileges, as demonstrated in Phases 2 and 3 of the AKC (Sections 3.2 and 3.3, respectively), where goal hijacking and lateral movement occur through authenticated channels [23].

We term this paradigm shift "Cognitive Authorization": requiring agents to generate *verifiable justifications* for high-risk actions before execution. Unlike identity-based authorization (proving «I am agent X with credential C»), cognitive authorization requires proving «This action serves objective O because of reasoning chain R, which is consistent with my task context and safe according to policy P.» The agent must externalize its intent in a form amenable to independent verification, addressing both the corruption of individual agent reasoning (Phase 2) and the propagation of malicious intent across agent networks (Phase 3).

This intent verification can be operationalized through structural patterns that enforce explicit externalization. For instance, Del Rosario et al.'s Plan-then-Execute pattern decouples the planning phase—where the agent reasons over potentially infected data—from the execution phase, forcing the agent to produce an immutable plan artifact that functions as a verifiable "contract" [33]. Once inspected and approved by human reviewers or automated policy checkers, the execution engine follows this plan deterministically, creating a verification gate that provides resilience against runtime deviations caused by indirect prompt injection.

To balance security with autonomy, Tallam and Miller propose extending this principle through tiered-risk access control [30]. By classifying agent actions into risk tiers (Green for low-risk retrieval, Yellow for monitored operations, Red for high-consequence actions), systems can apply proportional verification requirements. While Green-tier actions proceed autonomously, Red-tier actions trigger mandatory human-in-the-loop approval or require the agent to provide enhanced justification—such as explicit causal chains linking the proposed action to the user's original request. Although this oversight reintroduces latency, it represents a necessary trade-off for high-consequence actions in the current state of agentic AI.

For safety-critical domains requiring mathematical guarantees, these architectural controls can be augmented with formal methods. Zhang et al. argue that since probabilistic language models cannot eliminate hallucination, integration with formal verification is necessary to achieve provable plan safety [34]. In this paradigm, agents translate natural language plans into formal specifications (e.g., temporal logic formulas) that are mathematically verified against safety properties before execution. Ramani et al. demonstrate the practical feasibility of this approach, showing that advanced LLMs can generate syntactically correct formal models with high accuracy [35]. While the *formalization gap* and computational overhead currently restrict this approach to specific action sequences, it provides a deterministic assurance layer where the verifier rejects unsafe plans regardless of how convincingly the agent argues for them.

Ultimately, the unifying principle across these approaches—whether structural, human-overseen, or formal—is the requirement that agents externalize and justify their intent in verifiable forms. As anticipated, this effectively transforms the security question from «Is this agent authorized?» to «Is this action aligned with legitimate objectives?», providing a robust defence against cognitive compromise.

5.3. Circuit Breakers: Mechanical Limits on Autonomous Actions

If all upstream defences fail—the context is infected (Phase 1), reasoning is compromised (Phase 2), and malicious intent propagates across agents (Phase 3)—the final safeguard must be deterministic, mechanical constraints on what actions an agent can physically execute. We term this paradigm "Circuit Breakers": hard limits that trigger when agents approach safety boundaries, forcing human-in-the-loop intervention or system shutdown regardless of the agent's reasoning state or asserted justifications. Unlike Context Auditing (Section 5.1) or Intent Verification (Section 5.2), which operate on semantic signals and can potentially be fooled by sophisticated adversaries who craft convincing justifications, Circuit Breakers operate on *observable consequences*: measurable properties of actions that can be monitored and constrained without interpreting the agent's internal reasoning.

The paradigm's core principle is impact limitation through non-semantic constraints. Rather than attempting to understand *why* an agent is behaving anomalously, Circuit Breakers enforce simple rules based on observable metrics.

The aforementioned tiered-risk taxonomy proposed by Tallam and Miller [30] into Green, Yellow and Red (low-risk, monitored and high-consequence, respectively) is an example of a strategy that enables the systematic enforcement of mechanical constraints for Red-tier actions: cryptocurrency wallets reject transactions above monetary thresholds without multi-factor approval, file systems deny deletion on critical paths, and cloud APIs rate-limit provisioning to prevent denial-of-service scenarios. These limits operate at the actuator layer—the interface between agent decisions and real-world effects—creating a hard boundary independent of the agent's internal logic.

Theoretically, Circuit Breakers represent the conceptual inverse of the formal verification discussed in Section 5.2. While formal methods provide *pre-execution plan validation*, Circuit Breakers provide *post-hoc impact limitation*, constraining cumulative effects as they occur. Su et al. formalize this through a Constrained Markov Decision Process [12], modelling agent behaviour as a constrained optimization problem of the form: $\max_\pi \mathbb{E}[\text{Reward}]$ subject to $\mathbb{E}[\text{Risk}(\pi)] \leq \tau$ for some policy π and known constraint budget τ. In this model, Circuit Breakers implement the risk constraint as deterministic bounds that cannot be exceeded even by optimal adversarial policies. Where formal verification asks «Is this plan provably safe?», Circuit Breakers enforce «Even if this plan is unsafe, its maximum damage is bounded by X.»

The paradigm's ultimate strength lies in its resilience to cognitive attacks. Because Circuit Breakers do not interpret natural language, do not trust agent-provided justifications, and do not rely on reasoning about intent, they cannot be subverted by prompt injection or goal hijacking. An adversary who has achieved complete cognitive control of an agent network—compromising every model's reasoning and every protocol's trust assumptions—still cannot bypass mechanical limits enforced at the actuator hardware or API gateway level. This property makes Circuit Breakers the defensive paradigm of last resort, ensuring that even total cognitive compromise results in bounded rather than catastrophic impact.

These three paradigms—Context Auditing, Intent Verification, and Circuit Breakers—are not mutually exclusive but complementary layers in a defence-in-depth strategy aligned with the AKC framework. Context Auditing provides visibility into Phase 1 (what information enters the agent's perceptual boundary), Intent Verification validates reasoning integrity at Phases 2-3 (why actions are planned and how they propagate across agent networks), and Circuit Breakers enforce mechanical constraints at Phase 4 (what impacts can materialize in the physical or digital world). We do not claim these paradigms are exhaustive or that the specific implementations discussed (SHIELD, Plan-then-Execute, tiered-risk frameworks) are optimal for all deployment contexts. Instead, they represent a structured taxonomy for evaluating defensive mechanisms according to which cognitive phases they protect and what threat models they address. Alternative instantiations—adversarial robustness training at the model level, blockchain-based (eventually AI-enhanced [36]) immutable audit trails for decision provenance, cryptographic attestation of agent states, or zero-knowledge proofs for verifiable inter-agent communication—remain valid approaches that may instantiate these paradigms or constitute orthogonal defensive layers addressing attack vectors not captured by the AKC framework. Future research should systematically compare the effectiveness, overhead, and coverage of these diverse approaches across deployment scenarios, threat models, and autonomy levels to establish empirical foundations for agent security engineering.

6. Conclusion

The transition from the Web of Documents to the WoA represents one of the most significant architectural shifts in the history of distributed computing. As autonomous systems move from research prototypes to production deployments (managing financial transactions, controlling physical infrastructure, and making consequential decisions with minimal human oversight), the security challenges

they introduce can no longer be addressed through incremental adaptations of traditional cybersecurity frameworks.

This paper has introduced the **Agentic Kill Chain** (AKC), a threat modelling framework that reconceptualizes cyberattacks in the context of autonomous, LLM-powered agent ecosystems. Unlike conventional kill chains that model attacks as penetration through network and application layers, the AKC recognizes that in agent systems, the primary vulnerability lies in the *semantic interpretation layer*, the cognitive boundary where natural language is transformed into action. The framework's four phases (Semantic Infection, Cognitive Compromise, Agency Propagation, and Systemic Execution) map directly to the OODA loop stages, providing a structured approach to identifying vulnerabilities at each cognitive layer.

Through our analysis of modern agent protocols (MCP, A2A) and empirical attack vectors documented in recent literature, we have demonstrated that attacks on autonomous agents exhibit fundamentally different characteristics than attacks on traditional software:

- They target *interpretation* rather than implementation
- They propagate through *trust relationships* rather than network topology
- They manifest as *semantic contagion* rather than binary malware
- Their impact scales through *autonomous amplification* in multi-agent systems

We have identified three defensive paradigms aligned with the AKC's cognitive phases—Context Auditing (Phase 1), Intent Verification (Phases 2-3), and Circuit Breakers (Phase 4)—that represent necessary shifts in security thinking for autonomous agent ecosystems, moving from network-centric to cognitive-layer defences. These paradigms provide a structured framework within which concrete mechanisms can be evaluated and deployed according to deployment-specific threat models. However, we do not claim these paradigms are exhaustive, nor that specific instantiations discussed in Section 5 are optimal across all contexts. Proper security will require systematic integration with complementary approaches—adversarial robustness training, cryptographic attestation, formal verification, behavioural monitoring—and empirical validation of which mechanisms achieve optimal security-performance trade-offs across diverse deployment scenarios.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American 284 (2001) 34–43.

[2] T. Petrova, B. Bliznioukov, A. Puzikov, R. State, From Semantic Web and MAS to Agentic AI: A Unified Narrative of the Web of Agents, 2025. arXiv:2507.10644.

[3] S. Poslad, Specifying protocols for multi-agent systems interaction, ACM Trans. Auton. Adapt. Syst. 2 (2007) 15–es. doi:10.1145/1293731.1293735.

[4] G. F. Italiano, A. Martino, G. Piccardo, Invited Paper: Security and Privacy in Large Language and Foundation Models: A Survey on GenAI Attacks, in: Q. Bramas, B. Chatterjee, S. Devismes, M. Egan, P. S. Mandal, K. Mukhopadhyaya, V. V. Saradhi (Eds.), Distributed Computing and Intelligent Technology, Springer Nature Switzerland, Cham, 2025, pp. 1–17. doi:10.1007/978-3-031-81404-4_1.

[5] E. De Santis, A. Martino, A. Rizzi, Human Versus Machine Intelligence: Assessing Natural Language Generation Models Through Complex Systems Theory, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (2024) 4812–4829. doi:10.1109/TPAMI.2024.3358168.

[6] E. De Santis, A. Martino, E. Bruno, A. Rizzi, 2025: A GPT Odyssey. Deconstructing Intelligence by Gradual Dissolution of a Transformer, in: 2025 International Joint Conference on Neural Networks (IJCNN), 2025, pp. 1–10. doi:10.1109/IJCNN64981.2025.11227797.

[7] E. De Santis, A. Martino, F. Ronci, A. Rizzi, LSTM in Recursive Feedback Loops: A Study on Textual Evolution and Complexity, in: 2025 International Joint Conference on Neural Networks (IJCNN), 2025, pp. 1–10. doi:10.1109/IJCNN64981.2025.11227357.

[8] M. Xu, J. Fan, X. Huang, C. Zhou, J. Kang, D. Niyato, S. Mao, Z. Han, Xuemin, Shen, K.-Y. Lam, Forewarned is Forearmed: A Survey on Large Language Model-based Agents in Autonomous Cyberattacks, 2025. arXiv:2505.12786.

[9] Anthropic, Model Context Protocol (MCP) Specification, 2024. Available at: https://modelcontextprotocol.io.

[10] Google, Agent-to-Agent (A2A) Protocol Specification, 2025. Available at: https://a2aproject.github.io/A2A/latest/.

[11] F. P. B. Osinga, Science, strategy and war: The strategic theory of John Boyd, 1 ed., Routledge, 2007.

[12] H. Su, J. Luo, C. Liu, X. Yang, Y. Zhang, Y. Dong, J. Zhu, A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents, 2025. arXiv:2506.23844.

[13] S. Johnson, V. Pham, T. Le, Manipulating LLM Web Agents with Indirect Prompt Injection Attack via HTML Accessibility Tree, 2025. arXiv:2507.14799.

[14] R. Shahroz, Z. Tan, S. Yun, C. Fleming, T. Chen, Agents Under Siege: Breaking Pragmatic Multi-Agent LLM Systems with Optimized Prompt Attacks, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 9661–9674. doi:10.18653/v1/2025.acl-long.476.

[15] NVIDIA, Modeling Attacks on AI-Powered Apps with the AI Kill Chain Framework, NVIDIA Technical Blog, 2024. URL: https://developer.nvidia.com/blog/modeling-attacks-on-ai-powered-apps-with-the-ai-kill-chain-framework/, accessed: 2025-11-18.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017, pp. 1–11.

[17] E. Zverev, S. Abdelnabi, S. Tabesh, M. Fritz, C. H. Lampert, Can LLMs separate instructions from data? and what do we even mean by that?, in: The Thirteenth International Conference on Learning Representations, 2025, pp. 1–33. URL: https://openreview.net/forum?id=8EtSBX41mt.

[18] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz, Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection, in: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 79–90. doi:10.1145/3605764.3623985.

[19] J. Xu, J. W. Stokes, G. McDonald, X. Bai, D. Marshall, S. Wang, A. Swaminathan, Z. Li, AutoAttacker: A Large Language Model Guided System to Implement Automatic Cyber-attacks, 2024. arXiv:2403.01038.

[20] S. Datta, S. K. Nahin, A. Chhabra, P. Mohapatra, Agentic AI Security: Threats, Defenses, Evaluation, and Open Challenges, 2025. arXiv:2510.23883.

[21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[22] M. S. Salek, M. Chowdhury, M. B. Munir, Y. Cai, M. I. Hasan, J.-M. Tine, L. Khan, M. Rahman, A Large Language Model-Supported Threat Modeling Framework for Transportation Cyber-Physical Systems, IEEE Access 13 (2025) 163046–163070. doi:10.1109/ACCESS.2025.3603580.

[23] V. S. Narajala, O. Narayan, Securing Agentic AI: A Comprehensive Threat Model and Mitigation Framework for Generative AI Agents, 2025. arXiv:2504.19956.

[24] M. A. Ferrag, N. Tihanyi, D. Hamouda, L. Maglaras, M. Debbah, From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows, 2025. arXiv:2506.23260.

[25] Y. Huang, C. Wang, X. Jia, Q. Guo, F. Juefei-Xu, J. Zhang, Y. Liu, G. Pu, Efficient Universal Goal Hijacking with Semantics-guided Prompt Organization, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 5796–5816. doi:10.18653/v1/2025.acl-long.290.

[26] I. Kavathekar, H. Jain, A. Rathod, P. Kumaraguru, T. Ganu, TAMAS: Benchmarking Adversarial Risks in Multi-Agent LLM Systems, 2025. arXiv:2511.05269.

[27] D. Lee, M. Tiwari, Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems, 2024. arXiv:2410.07283.

[28] P. He, Y. Lin, S. Dong, H. Xu, Y. Xing, H. Liu, Red-Teaming LLM Multi-Agent Systems via Communication Attacks, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Findings of the Association for Computational Linguistics: ACL 2025, Association for Computational Linguistics, Vienna, Austria, 2025, pp. 6726–6747. doi:10.18653/v1/2025.findings-acl.349.

[29] A. Toggi, B. Bose, D. Naidu, R. Srivastava, Metasploit based automated penetration testing using reinforcement learning, in: 2024 First International Conference for Women in Computing (InCoWoCo), 2024, pp. 1–8. doi:10.1109/InCoWoCo64194.2024.10863399.

[30] K. Tallam, E. Miller, Operationalizing CaMeL: Strengthening LLM Defenses for Enterprise Deployment, 2025. arXiv:2505.22852.

[31] T. Xu, Z. Wen, X. Zhao, J. Wang, Y. Li, C. Liu, L2M-AID: Autonomous Cyber-Physical Defense by Fusing Semantic Reasoning of Large Language Models with Multi-Agent Reinforcement Learning, 2025. URL: https://arxiv.org/abs/2510.07363. arXiv:2510.07363.

[32] Y. Hmimou, M. Tabaa, A. Khiat, Z. Hidila, A Multi-Agent System for Cybersecurity Threat Detection and Correlation Using Large Language Models, IEEE Access 13 (2025) 150199–150215. doi:10.1109/ACCESS.2025.3602681.

[33] R. F. Del Rosario, K. Krawiecka, C. S. de Witt, Architecting Resilient LLM Agents: A Guide to Secure Plan-then-Execute Implementations, 2025. arXiv:2509.08646.

[34] Y. Zhang, Y. Cai, X. Zuo, X. Luan, K. Wang, Z. Hou, Y. Zhang, Z. Wei, M. Sun, J. Sun, J. Sun, J. S. Dong, Position: Trustworthy AI agents require the integration of large language models and formal methods, in: A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, J. Zhu (Eds.), Proceedings of the 42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, PMLR, 2025, pp. 82441–82459.

[35] K. Ramani, V. Tawosi, S. Alamir, D. Borrajo, Bridging LLM Planning Agents and Formal Methods: A Case Study in Plan Verification, 2025. arXiv:2510.03469.

[36] D. Ressi, R. Romanello, C. Piazza, S. Rossi, Ai-enhanced blockchain technology: A review of advancements and opportunities, Journal of Network and Computer Applications 225 (2024) 103858. doi:10.1016/j.jnca.2024.103858.