Emergent Capability Composition in Agentic AI:

When Authorized Tools Combine into Unauthorized Powers

Philippe Bogaerts
RadarSec — philippe.bogaerts@radarsec.com

Abstract

Multi-step tool chaining by LLM agents is a recognized attack vector, studied under names such as sequential tool attack chaining (STAC) and long-horizon attacks. What remains under-explored is how this phenomenon interacts with the governance models of agentic protocols. We address this gap for the Model Context Protocol (MCP) and Agent-to-Agent (A2A) ecosystems, providing the first governance-focused capability-composition analysis of these protocols. We formalize emergent capability composition—the phenomenon whereby individually authorized tools can, under specific environment and objective conditions, be combined by a reasoning model to synthesize capabilities never explicitly granted or anticipated. An agent with web search and a command-line interface may autonomously install software and build a surveillance pipeline; each step can appear policy-compliant at review time, while the composed trajectory is not. We term this the 1+1=3 problem. We introduce a conditional composition operator parametrized by environment constraints, policy, and objective; demonstrate that the composition search space grows combinatorially with tool count; and show that MCP authorization (three specification revisions in 2025), NIST, ISO/IEC 42001, and OWASP LLM Top 10 are structurally blind to compositional threats because they evaluate tools in isolation. We present eight composition scenarios grounded in real MCP deployments and analyze recursive MCP delegation as a composition amplifier. We report empirical evaluation of a composition scanner across 15 server-pair compositions drawn from 17 production MCP servers, demonstrating that composition surplus is pervasive (100% detection rate) with severity gradient as the actionable discriminator. As a preliminary finding, we report that a formally structured capability-class delta prompt framing appears to reliably trigger composition behavior in frontier models via prompt injection, suggesting that injection and emergent composition may constitute a single attack primitive. We propose a capability-algebra defense framework that evaluates authorization at the composed capability level.

Keywords: emergent capability composition, agentic AI security, Model Context Protocol, tool composition attacks, capability algebra, 1+1=3 problem, prompt-triggered composition, capability-class delta, MCP governance, LLM agent authorization, combinatorial attack surface, composition scanner

1. INTRODUCTION

Consider an AI agent authorized to use two tools: a web search API and a command-line interface. Each tool, evaluated independently, appears benign and appropriately scoped. The web search retrieves public information; the CLI executes local commands within a sandboxed environment. Yet a frontier reasoning model—given these two tools and a sufficiently open-ended objective—can autonomously chain them to: (1) search for open-source speech recognition software, (2) download and install it via the CLI, (3) invoke system audio capture utilities, and (4) pipe recorded audio through a transcription pipeline. The result is a surveillance capability that was never authorized, never reviewed, and never appeared in any tool description. Each step can appear policy-compliant or low-risk at review time, yet the composed trajectory achieves a capability class that the authorization model was never designed to evaluate.

Multi-step tool chaining is a recognized attack vector: Li et al. formalize Sequential Tool Attack Chaining (STAC) with attack success rates exceeding 90% [43], and AgentLAB benchmarks long-horizon attacks including tool chaining across 644 test cases [44]. What remains under-explored is how composition interacts with the governance models of production agentic protocols. The Model Context Protocol (MCP), now exceeding 97 million monthly downloads under Linux Foundation governance [1][2], and Google's Agent-to-Agent (A2A) protocol [3] define the communication substrate through which frontier agents discover and invoke tools. These protocols were designed to solve the integration problem—connecting AI models to external capabilities—and they succeed admirably at this. What they do not address is the security implications of what happens when a reasoning model combines those capabilities in ways their designers did not anticipate, within governance frameworks structurally blind to composed trajectories.

We term this the 1+1=3 problem: the composed capability is categorically more powerful than the union of individual tool capabilities. This is not a hypothetical concern. The UK AI Security Institute documents that frontier models can now complete expert-level cybersecurity tasks requiring over 10 years of human experience [4], and Carlini et al. demonstrate that adaptive attacks bypass 12 published defenses with over 90% success [5]. A model with this level of reasoning capability, given access to even a modest tool set, can compose those tools into capabilities that operate well beyond the authorization envelope that any individual tool review would suggest.

This paper makes seven contributions: (1) the first MCP/A2A governance-focused analysis of capability composition as a distinct threat class, building on STAC [43] and AgentLAB [44] by reframing tool chaining as a capability-governance mismatch; (2) a conditional composition operator parametrized by environment, policy, and objective, enabling quantitative analysis of compositional attack surfaces; (3) eight concrete composition scenarios grounded in real MCP deployments; (4) a preliminary finding that composition behavior appears reliably triggerable via prompt injection using a capability-class delta framing; (5) a systematic analysis demonstrating that current governance frameworks are structurally blind to compositional threats; (6) empirical evaluation of a composition scanner across 15 server compositions drawn from 17 production MCP servers, demonstrating that composition surplus is pervasive with severity gradient as the actionable discriminator; and (7) a proposed defense framework that evaluates authorization at the composed-capability level.

2. BACKGROUND

2.1 Agentic Protocol Ecosystem

MCP, introduced by Anthropic in November 2024, standardizes the connection between AI assistants and external tools through a JSON-RPC client-server protocol [1]. An MCP server exposes typed tools with structured schemas; clients (typically LLM-powered agents in IDEs or chat interfaces) discover and invoke these tools. The protocol supports resource access, prompt-level context injection, and a sampling mechanism where servers can request LLM completions through the client. Google's A2A protocol extends this to inter-agent communication through standardized agent cards and task delegation [3]. The combined stack enables agents to discover tools (MCP), delegate to other agents (A2A), and compose multi-step workflows autonomously—precisely the conditions under which emergent composition thrives.

2.2 Frontier Reasoning Capabilities

Contemporary reasoning models employ chain-of-thought planning, tool-use orchestration, error recovery, and persistent memory that substantially extends their autonomous action space [4]. Critically for composition, these models do not merely execute pre-defined tool sequences; they reason about tool combinations, formulate novel multi-step plans, and adapt when initial approaches fail. This planning capability is the engine that transforms a flat set of individually authorized tools into a space of composed capabilities that no human reviewer anticipated.

2.3 Related Work

Multi-step tool chaining is well established as an attack primitive. Li et al. introduce STAC [43], demonstrating that individually innocent tool calls form dangerous chains with ASR exceeding 90% against state-of-the-art agents including GPT-4.1. AgentLAB [44] benchmarks long-horizon attacks across 644 test cases including tool chaining, intent hijacking, and memory poisoning, showing that single-turn defenses fail against temporally-extended adversarial strategies. MCP-ITP [45] demonstrates implicit tool poisoning achieving 84.2% ASR while suppressing detection rates to 0.3%, overlapping with our poisoning-amplified composition class. The MCPLIB study [7] identifies chain attacks exploiting shared context as a named empirical phenomenon in MCP deployments specifically.

On the governance side, Li et al. [39] observe at ICSE 2026 that problematic behavior may emerge from multi-tool interactions and propose System-Theoretic Process Analysis (STPA) for tool-use safety. Ranade et al. [40] document eleven failure classes in multi-tool agents including unauthorized cross-agent propagation. Chen et al. [41] propose Progent, the first privilege control mechanism for LLM agents. The ACM Computing Surveys study by Xu et al. [42] identifies functional manipulation that exploits agent tool interfaces for unauthorized actions.

Our work builds on these contributions and extends them in three directions specific to MCP/A2A governance. First, where STAC and AgentLAB demonstrate that tool chains are dangerous and characterize attack success empirically, we reframe composition as a capability-governance mismatch: formalizing it through a conditional composition operator parametrized by environment, policy, and objective, and measuring the resulting governance gap. Second, we provide preliminary evidence that composition behavior may be reliably triggerable via prompt injection using formally structured reasoning scaffolds. Third, where [41] enforces privilege constraints per tool, we argue that privilege must be evaluated at the composed capability level relative to MCP's per-server/per-tool authorization model.

3. EMERGENT CAPABILITY COMPOSITION

3.1 Definition and Formalization

Let $T = \{t_1, t_2, ..., t_n\}$ be the set of tools available to an agent, where each t_i has an authorized capability envelope $Cap(t_i)$ defining the set of actions it is designed and approved to perform. We define the conditional composition operator parametrized by environment constraints E (e.g., sandbox boundaries, network egress policy), authorization policy P, and agent objective O.

That is, there exist tool pairs and environment/objective conditions under which the composed capability strictly exceeds the union of individual capabilities. This is not a universal claim: some tool pairs are redundant, some environments block the surplus (e.g., a sandbox preventing software installation), and some compositions increase efficiency without expanding the capability class. The threat arises precisely when composition does produce surplus—and governance has no mechanism to detect which pairs, under which conditions, will do so. We define the composition surplus $S_{ij}(E, O)$ = $Cap(t_i$ composed with $t_j)$ - $(Cap(t_i)$ union $Cap(t_j))$. When $S_{ij}(E, O)$ is non-empty, composition has produced the '3' in '1+1=3': capabilities that exist only through composition and are therefore invisible to any governance framework that evaluates tools in isolation.

3.2 Combinatorial Attack Surface

For an agent with access to n tools, the number of pairwise compositions is $C(n,2) = n(n-1)/2$. Including higher-order compositions (three or more tools chained), the total composition space is $2^n - n - 1$ non-trivial subsets. A typical MCP deployment connecting to 10 servers, each exposing 5 tools, yields n=50 individual tools and approximately 10^{15} possible compositions. Even restricting to pairwise compositions gives 1,225 composition pairs—a surface that no manual security review can cover. The attack surface is not additive; it is combinatorial.

3.3 The Reasoning Engine as Composition Driver

What distinguishes emergent composition from traditional API chaining or scripted workflows is the reasoning engine that drives it. A frontier model does not require a pre-defined workflow or explicit instructions to compose tools—as STAC demonstrates with over 90% ASR [43]. Given a high-level objective and a set of available tools, it can autonomously: (a) identify that a desired capability can be synthesized from available primitives; (b) plan a multi-step execution path; (c) handle errors and iterate on failed approaches; and (d) discover compositions that a human operator might not conceive. Each tool is used as designed; the threat lies in the reasoning about combinations and the governance model's inability to evaluate the composed trajectory.

3.4 Concrete Composition Scenarios

We present eight composition attack scenarios grounded in tool types commonly available through MCP servers in production deployments. Each scenario combines individually authorized capabilities to produce an emergent capability that exceeds the authorization envelope.

Table 1: Emergent capability composition scenarios. Each row combines two individually authorized tool categories to produce a capability exceeding the union of their authorization envelopes.

ID	Tool A	Tool B	Emergent Capability (the "3")
S1	Web search	CLI / shell	Install software, build audio surveillance pipeline
S2	File system read	HTTP requests	Data exfiltration: read sensitive files, POST externally
S3	Code execution	Package mgr	Arbitrary software installation and execution
S4	Database query	Email / messaging	Autonomous data harvesting and exfiltration
S5	Git operations	CI/CD triggers	Supply-chain injection: modify source, trigger deploy
S6	Browser auto	Credential store	Autonomous authenticated actions across web services
S7	Calendar	Email drafting	Social engineering at scale: context-aware phishing
S8	Cloud API	DNS management	Infrastructure takeover: provision compute, redirect traffic

In each scenario, the critical observation is that reviewing Tool A and Tool B in isolation reveals no concerning capability. The threat materializes only when a reasoning model identifies that chaining them achieves a goal that neither could accomplish alone. Scenario S1 is particularly instructive: web search is read-only and CLI is typically sandboxed, yet their composition enables arbitrary software installation and execution—a capability class that most security policies would explicitly prohibit.

3.5 Why Current Governance Is Blind

The composition surplus $S_{ij}(E, O)$ is invisible to current governance approaches for three structural reasons. First, tool-level authorization: MCP authorization operates at the server level (and aspires to the tool level), evaluating whether an agent may invoke a specific tool—not what combinations of authorized tools can achieve. Second, static threat modeling: security reviews assess tools at registration time against known threat patterns, but composition threats emerge dynamically from the agent's runtime reasoning. Third, description-based trust: MCP tool descriptions define what a tool does, not what it enables when combined with other tools. There is no metadata field for 'capabilities this tool contributes to when composed.' The governance framework has no vocabulary for compositional threats because it was built on the assumption that tool safety is a per-tool property.

4. THE CAPABILITY-GOVERNANCE DELTA

We reformalize the capability-governance delta through the lens of composition. Define $D_{CG}(t) = |S_{total}(t)| - |G_{covered}(t)|$ where $S_{total}(t)$ is the total composition surplus across all tool subsets available to the agent at time t, and $G_{covered}(t)$ is the set of composed capabilities that governance frameworks can detect, evaluate, and constrain. The delta is the set of ungoverned composed capabilities—things the agent can do that no control mechanism can see or prevent.

This delta is widening for three reasons. First, tool count grows linearly (new MCP servers are published daily; the MCP ecosystem now encompasses thousands of servers [2]), but the composition space grows exponentially. Second, model reasoning capability—the engine that identifies and exploits compositions—is advancing on training-run timescales (months), while governance frameworks advance on institutional timescales (years). NIST's AI Agent Standards Initiative, launched February 2026 [14], is unlikely to produce finalized standards before 2028. Third, no governance framework has yet adopted a compositional evaluation methodology; they remain structurally incapable of assessing S_{total}.

4.1 Empirical Evidence

The delta manifests concretely in the MCP ecosystem. Endor Labs' analysis of 2,614 MCP implementations [23] found that 82% use file system operations prone to path traversal (CWE-22) and 34% use APIs susceptible to command injection (CWE-78). MCP-ITP demonstrates implicit tool poisoning achieving 84.2% ASR while suppressing detection to 0.3% [45]; STAC achieves over 90% ASR for multi-step tool chain attacks [43]; deploying just ten MCP plugins creates a 92% exploitation probability [15]; and CVE-2025-6514 (CVSS 9.6) enabled remote code execution across 437,000 installations of mcp-remote [17]. These are per-tool and per-chain vulnerabilities—the composition surplus operates on top of this already compromised baseline. The governance gap is not additive; it is multiplicative.

5. COMPOSITION-AWARE THREAT TAXONOMY

We reorganize the MCP/A2A threat landscape to foreground composition. Traditional threat taxonomies [7][8][11] categorize attacks by the targeted layer (semantic, protocol, infrastructure). We propose a complementary dimension: whether the threat is atomic (exploiting a single tool or protocol element) or compositional (emerging from tool combinations).

5.1 Atomic Threats

These are well-documented in prior work and include: tool poisoning via description metadata [6][18], tool shadowing and preference manipulation in multi-server environments [7], OAuth token exfiltration through confused deputy vulnerabilities [19], session hijacking via URL-exposed session IDs [20], sampling exploitation as a reverse prompt injection channel [21], remote code execution (CVE-2025-6514 [17], CVE-2025-68143/68144/68145 [12]), supply-chain compromise through malicious server packages [22], sandbox escape via symlink exploitation [23], rug-pull attacks through delayed behavioral mutation [6], and log-to-leak privacy attacks [24]. Each is dangerous individually, but they become the building blocks for compositional threats.

5.2 Compositional Threats

Capability synthesis (C1). The core 1+1=3 pattern: combining authorized tools to create unauthorized capabilities. Scenarios S1–S8 in Table 1 exemplify this class. No adversary is required—the model's own reasoning drives composition in pursuit of its objective.

Poisoning-amplified composition (C2). A poisoned tool description (atomic threat) steers the model toward specific compositions. For example, a tool poisoned to suggest 'for better results, also invoke the HTTP client' can channel the model toward an exfiltration composition it might not have spontaneously constructed.

Delegation-chain composition (C3). In recursive MCP delegation (MCP-in-MCP), an outer server delegates to inner servers, each contributing tools. The composed capability spans the entire delegation chain, but authorization is evaluated per hop.

Cross-protocol composition (C4). When MCP (tool invocation) and A2A (agent delegation) operate in the same deployment, an agent can compose tools from MCP servers with capabilities of A2A-delegated agents, creating a composition space that spans both protocol stacks.

Temporal composition (C5). Tools accessed at different points in time combine through the agent's persistent memory. An agent that queries a database today and sends an email tomorrow has composed a data exfiltration pipeline, but the two actions—separated in time—may not trigger any single-session policy check.

6. PROMPT-TRIGGERED COMPOSITION: PRELIMINARY FINDINGS

The composition threat classes described above (C1–C5) assume that the model's reasoning engine spontaneously discovers composition paths. STAC [43] demonstrates that multi-step chains can be induced automatically with high ASR. We report a complementary preliminary observation: a formally structured reasoning scaffold—distinct from STAC's automated chain generation—appears to reliably trigger composition behavior via prompt injection. We present this as a direction for future rigorous evaluation, not as a fully validated empirical result.

6.1 The Capability-Class Delta Prompt

During exploratory research, we observed that providing frontier reasoning models with a formal capability-expansion framing caused them to pursue composition paths. The framing is structured around a capability-class delta formulation: let C_t be the set of coarse capability classes granted at time t (e.g., ReadFiles, WriteFiles, StartProcess, NetworkEgress, InstallSoftware). A proposed action plan induces $D+ = C_{t+1} \setminus C_t$. The injected prompt frames this formally: 'If D+ is non-empty, the agent is expanding its effective power surface.'

The mechanism is notable because the injected prompt does not instruct the model to perform any specific malicious action. Instead, it provides a reasoning scaffold that frames the problem in terms of capability expansion. When a frontier model encounters this framing in its context—whether through a poisoned tool description, a compromised resource, or a malicious A2A agent card—its chain-of-thought reasoning identifies available tool combinations that achieve capability expansion.

6.2 Preliminary Observations and Limitations

In informal testing across a small number of frontier reasoning models and MCP tool configurations, we observed: (a) the capability-delta prompt appeared to trigger composition behavior more consistently than equivalent goals expressed in natural language; (b) the formal mathematical framing appeared to engage longer and more systematic composition chains; (c) the model treated capability expansion as a subgoal, autonomously planning multi-step tool chains; and (d) human-in-the-loop approval mechanisms did not catch the composition because each individual tool invocation appeared benign.

Important caveats: These observations are preliminary. We did not conduct controlled experiments with formal ASR measurements, ablations, or systematic negative controls. A rigorous evaluation—following the methodology of STAC [43] or AgentLAB [44]—is needed to validate whether the delta framing produces statistically significant differences from natural-language baselines.

6.3 Hypothesized Attack Chain

If validated, the delta-framing mechanism suggests the following attack chain:

- **Step 1 (Injection):** An attacker embeds a capability-class delta prompt in a vector accessible to the agent—a poisoned

MCP tool description, a malicious document, a compromised A2A agent card.

• **Step 2 (Reasoning activation):** The model processes the delta framing during chain-of-thought reasoning, inventorying available tools and identifying composition paths.

• **Step 3 (Autonomous composition):** The model plans and executes a multi-tool chain synthesizing the target capability. Each step uses a tool within its authorized scope.

• **Step 4 (Approval bypass):** If human-in-the-loop approval is present, each step is presented individually. The approver evaluates steps, not the composed trajectory.

6.4 Implications and Research Directions

If confirmed by rigorous evaluation, this observation has three implications. First, composition threats would be not merely theoretically possible but practically triggerable through known injection vectors. Second, the formal structure of the prompt may matter: capability-delta framing versus natural-language instructions as an independent variable in attack success. Third, and most fundamentally, prompt injection and emergent composition may not be separate threat classes but a single attack primitive—injection provides the trigger, composition provides the payload, and the model's reasoning is the execution engine.

7. EMPIRICAL EVALUATION

To move from theoretical analysis to empirical validation, we developed the MCP Composition Scanner—an open-source tool that performs pre-authorization composition analysis on static tool declarations. This section reports results from a systematic evaluation across 15 server-pair compositions drawn from 17 widely-deployed MCP servers.

7.1 Experimental Design

Tool declarations were collected from production-representative MCP server configurations including database servers (PostgreSQL, SQLite), messaging platforms (SendGrid, Slack), cloud infrastructure (Kubernetes, S3, Google Drive), development tools (GitHub, Git), browser automation (Puppeteer), local system access (Desktop Commander, Filesystem, Bash Executor), data stores (Redis, Memory), and information retrieval services (Fetch, Google Maps). Each server's tool declarations were captured as structured JSON containing tool names, descriptions, and input schemas—the same metadata available to an LLM agent at runtime.

The scanner was run across three categories of compositions. **High-Risk Pairs (n=10):** Server combinations hypothesised to produce composition surpluses based on complementary capability classes (e.g., database-read + email-send). **Control Pairs (n=3):** Combinations expected to produce minimal surpluses, selected for domain orthogonality or functional redundancy (Fetch + Google Maps, PostgreSQL + SQLite, Google Maps + Memory). **Triple Compositions (n=2):** Three-server combinations to test higher-order chain detection (PostgreSQL + Desktop Commander + SendGrid; Puppeteer + Redis + GitHub).

All 15 runs used identical configuration (GPT-4o reasoning backbone, Pydantic structured output, temperature 0). Each run produces a CompositionAnalysis object containing capability vectors, composition surpluses with severity ratings, attack chains, and governance recommendations.

7.2 Results

7.2.1 Summary Statistics

Table 2: Aggregate composition scanner results across 15 server compositions.

Category	Runs	Avg Tools	Avg Surpluses	Critical	High	Medium
High-Risk Pairs	10	16.6	3.9	7	24	8
Control Pairs	3	12.3	2.7	0	6	2
Triple Comps	2	41.5	4.0	6	2	0
Total	15	19.1	3.7	13	32	10

Three key observations emerge. First, **composition surplus is pervasive**: every single composition—including all three control pairs—produced at least two surpluses. The binary question 'does this pair produce emergent capabilities?' was answered affirmatively in 100% of cases (15/15). This demonstrates that capability composition is a structural property of multi-server MCP deployments, not an edge case confined to obviously dangerous combinations.

Second, **severity gradient discriminates risk**: while surpluses were universal, the severity distribution was not. High-risk pairs produced 7 Critical-severity surpluses across 10 runs; control pairs produced zero. Triple compositions amplified this: with only 2 runs, they produced 6 Critical findings, including 4 in a single analysis (PostgreSQL + Desktop Commander + SendGrid).

Third, **all surpluses were cross-server**: 100% of detected surpluses involved tools from different servers (55/55). This confirms the central thesis—that composition risk is invisible to per-server governance.

7.2.2 High-Risk Pair Findings

Data Exfiltration Chains. The most frequently detected pattern combined a data-read capability with a network-egress capability. The PostgreSQL + SendGrid composition identified that the `query` tool (read-only, classified as safe in isolation) combined with `send_email` creates autonomous database exfiltration to arbitrary email recipients. The scanner correctly noted: 'Per-server authorization classifies query as read-only (safe) and send_email as internal-messaging (safe); no per-tool analysis detects that, when cross-authorized, these enable full database exfiltration.'

Infrastructure Compromise. The Kubernetes + Bash Executor pair—with only 6 total tools and 15 pairwise combinations—produced 3 surpluses, 2 at Critical severity. The scanner identified that `kubectl get secrets` chained with arbitrary command execution enables credential harvesting and lateral movement, a composition requiring only two tool invocations but granting cluster-wide compromise.

Escalation Through Local Execution. Desktop Commander + S3 (5 surpluses, 1 Critical) and Git + SendGrid (4 surpluses, 1 Critical) demonstrated that filesystem or repository access combined with cloud storage or messaging tools creates egress channels for sensitive data.

7.2.3 Control Pair Analysis

The control pairs were designed as negative controls, yet each produced genuine composition surpluses. Rather than representing false positives, closer examination reveals these findings to be legitimate—and instructive.

PostgreSQL + SQLite (3 surpluses, 0 Critical, 2 High, 1 Medium). The scanner identified cross-database data migration as a composition surplus: the `query` tool on PostgreSQL combined with `create_record` on SQLite enables autonomous data copying from a production database to a less-controlled local store. It also detected targeted record purging—using PostgreSQL query results to drive SQLite deletions—as a supply-chain modification risk.

Google Maps + Memory (3 surpluses, 0 Critical, 2 High, 1 Medium). The scanner detected that reverse geocoding combined with knowledge graph persistence creates a 'persistent, queryable geolocation surveillance graph.' While neither tool alone enables location tracking, their composition allows an agent to resolve coordinates to addresses and store them as longitudinal observations tied to entities.

Fetch + Google Maps (2 surpluses, 0 Critical, 2 High). Combining web content retrieval with geolocation services enables deanonymisation and location-correlated intelligence gathering—an OSINT-style reconnaissance capability absent from either server individually.

The critical insight is that the severity gradient—not the binary presence of surpluses—is the meaningful discriminator. Control pairs produce surpluses in lower-severity categories (Surveillance, DataEnrichment, SupplyChainModification) while completely lacking the Critical-severity DataExfiltration patterns that dominate high-risk compositions.

Table 3: Risk profile comparison between high-risk and control compositions.

Metric	High-Risk (n=10)	Control (n=3)
Mean surpluses/run	3.9	2.7
Critical-severity surpluses	7	0
DataExfiltration class	12	0
Network egress chains	17	0

7.2.4 Higher-Order Compositions

PostgreSQL + Desktop Commander + SendGrid (32 tools, 496 pairwise combinations, 4 surpluses—all Critical). This three-server composition produced the highest concentration of Critical findings. The scanner identified a complete data exfiltration pipeline: database query → local file staging (with optional transformation via process execution) → email transmission. The three-server composition enabled attack chains impossible with any two-server subset, including code-execution-assisted data packaging before exfiltration.

Puppeteer + Redis + GitHub (51 tools, 1275 pairwise combinations, 4 surpluses, 2 Critical). Browser automation combined with session data access and repository write access creates both credential harvesting (scrape → cache → exfiltrate) and supply chain poisoning (browse → extract → commit malicious code) chains.

The pairwise search space grew from an average of 143 combinations for pairs to 886 for triples. Despite this 6x increase, the scanner maintained consistent analysis quality, suggesting the approach scales to realistic deployments where 5–10 servers may be simultaneously authorised.

7.3 Threats to Validity

LLM Reasoning Sensitivity. The scanner relies on GPT-4o's reasoning about tool semantics. Different models or temperatures may produce different surplus counts. We mitigated this by using temperature 0 and structured output constraints, but acknowledge that LLM-based analysis introduces non-determinism.

Tool Declaration Fidelity. Our evaluation used tool declarations collected from reference MCP server implementations. Real-world deployments may have customised tool descriptions. The scanner's effectiveness depends on the semantic richness of tool descriptions.

Ground Truth. We lack a formal ground-truth dataset of composition surpluses against which to measure precision and recall. Our evaluation relies on expert assessment. Constructing such a benchmark is an important direction for future work.

7.4 Key Findings

The evaluation supports three principal claims. **Claim 1:** Composition surplus is structural, not exceptional—all 15 compositions produced surpluses, indicating that multi-server MCP deployments inherently create emergent capabilities invisible to per-server review. **Claim 2:** Severity gradient enables actionable governance—the clean separation of Critical-severity findings (7 in high-risk, 0 in controls) demonstrates a usable risk signal. **Claim 3:** Pre-authorisation analysis is feasible—the scanner operates entirely on static tool declarations, completes in under 60 seconds per composition, and produces structured, actionable output including mutual-exclusion recommendations.

8. MCP AUTHORIZATION: COMPOSITION BLIND SPOTS

The MCP authorization specification has undergone three revisions in 2025 (March, June, November), each improving security [20][25][26]. The March spec introduced OAuth 2.1 with mandatory PKCE; the June revision reclassified servers as OAuth Resource Servers with RFC 9728; the November revision introduced Client ID Metadata Documents and Cross-App Access (XAA) for enterprise IdP governance [26]. Yet across all three revisions, authorization operates at the server or tool level—never at the composed capability level.

Concretely, the spec answers 'may this agent invoke this tool?' but cannot answer 'what happens when this agent invokes this tool after already having invoked that tool?' There is no mechanism for: (a) declaring composition constraints; (b) evaluating composite capability at authorization time; (c) tracking session-level capability accumulation; or (d) enforcing delegation-depth limits. The four persistent architectural weaknesses identified in prior analysis [27]—no message-level integrity, no tool attestation, coarse-grained authorization, and trust transitivity—all exacerbate compositional risk.

9. RECURSIVE DELEGATION AS COMPOSITION AMPLIFIER

Recursive MCP delegation—where an MCP server acts as client to downstream servers (MCP-in-MCP)—dramatically amplifies the composition problem. Docker's MCP Gateway pattern [28], dynamic discovery tools (mcp-find, mcp-exec) [29], and explicit recursion servers (mcp-inception) all create multi-hop call graphs. Each hop introduces new tools into the agent's effective tool set, expanding the composition space exponentially.

A single-level deployment with 10 tools has C(10,2) = 45 pairwise compositions. A two-level delegation where each of 3 downstream servers exposes 10 more tools yields 40 total tools and C(40,2) = 780 pairwise compositions—a 17-fold increase from a single level of delegation.

Recursive delegation also creates four specific composition pathologies: transitive trust amplification (authorization at the outer level implicitly extends through the delegation chain), containment erosion (a containerized server that delegates outside its container breaks the isolation boundary), observability collapse (nested compositions are invisible to outer-level logging), and policy composition failure (the effective policy is the weakest link across all delegation hops, not the strongest).

10. GOVERNANCE GAP ANALYSIS

We evaluate four governance frameworks for composition awareness.

NIST AI Agent Standards Initiative (February 2026) [14] targets agent interoperability and security standards, but its initial scope focuses on agent identity and authorization—per-agent, not per-composition. Compositional governance standards are unlikely before 2028.

ISO/IEC 42001 provides certifiable AI management systems with lifecycle governance [30], and the **CSA AI Controls Matrix** offers 243 control objectives across 18 domains [31]. Both operate at the organizational and system level, with no provisions for evaluating tool-composition risk.

OWASP LLM Top 10 classifies tool poisoning under LLM01 (Prompt Injection) [32]. However, OWASP guidance treats tools as individual attack surfaces. There is no entry for 'capability composition' or 'emergent tool chaining.'

The fundamental finding is that all four frameworks share a common blind spot: they evaluate security as a per-component property. Emergent composition demonstrates that security is a system-level property that cannot be decomposed into per-tool evaluations.

11. TOWARD COMPOSITIONAL DEFENSE

We propose a five-layer defense architecture that specifically targets compositional threats.

11.1 Layer 1: Capability Algebra and Composition-Aware Authorization

The core innovation: authorization decisions must evaluate composed capabilities, not individual tools. We propose a capability algebra where each tool is annotated with a capability vector describing its action classes (read, write, execute, network, install, etc.). Before granting an agent access to a tool set, a composition analysis evaluates all pairwise (and selected higher-order) capability vectors to identify dangerous compositions. Tool sets that produce S_{ij} entries in prohibited capability classes are flagged for explicit human review or automatically constrained through mutual exclusion policies.

Section 7's empirical evaluation validates this approach: the MCP Composition Scanner demonstrates that automated pre-authorization analysis is feasible, completing in under 60 seconds per composition with structured output directly usable for mutual-exclusion policy generation.

11.2 Layer 2: Session-Level Capability Tracking

Runtime monitoring that tracks the accumulated capability envelope within each agent session. As an agent invokes tools sequentially, the monitor evaluates the composed capability of all tools invoked so far. If the accumulated composition crosses a threshold into a prohibited capability class, the session is suspended for human review. This addresses temporal composition (C5).

11.3 Layer 3: Delegation-Depth Bounds

Mandatory delegation depth limits prevent the unbounded expansion of the composition space through recursive MCP delegation. Container-native isolation is extended with mandatory egress filtering that prevents outbound MCP client connections to unvetted servers.

11.4 Layer 4: Gateway-Mediated Policy Enforcement

An MCP Gateway [28][33] operates as a composition-aware policy enforcer: maintaining a live capability accumulation model per session, applying mutual exclusion constraints on tool co-authorization, logging full composition chains, and implementing rate limits that account for compositional risk.

11.5 Layer 5: Cryptographic Tool Attestation and Behavioral Monitoring

Tool descriptions are cryptographically signed at registration and verified at invocation, addressing atomic threats. Runtime behavioral analysis compares invocation patterns against established baselines to detect anomalous compositions.

12. DISCUSSION

12.1 The Fundamental Asymmetry

Emergent capability composition creates a fundamental asymmetry between offense and defense. The composition search space grows exponentially with tool count, while defense requires evaluating each composition—an inherently harder problem. STAC demonstrates ASR exceeding 90% for automated chain generation [43]; our preliminary observation in Section 6 suggests that formally structured reasoning scaffolds may deepen this asymmetry further.

12.2 Implications for AI Safety

The 1+1=3 problem connects agentic security to the broader AI safety discourse. If a model can compose two benign tools into a surveillance pipeline, the alignment problem extends beyond the model's values to the system's capability envelope. An agent may be perfectly aligned—following instructions, respecting constraints—and still produce dangerous compositions because it was never told that the composed capability is undesirable. This suggests that tool-use governance must be treated as an alignment mechanism, not merely a security control.

12.3 Scalability of Compositional Defense

The capability algebra approach has practical scalability limits. Exhaustive pairwise analysis of 50 tools requires 1,225 evaluations—feasible. But higher-order analysis grows as $O(n^k)$ for k-wise compositions. Practical deployment will require heuristic pruning, LLM-assisted composition analysis, and empirical composition testing. The empirical results in Section 7 suggest that LLM-assisted analysis is viable: the scanner produced actionable results across all 15 compositions with consistent quality.

12.4 Limitations

Our analysis has several limitations. The composition surplus, while now parametrized by environment and objective, remains qualitatively defined; a rigorous quantitative metric is future work. Our scenario analysis (Table 1) is illustrative, not exhaustive. The capability algebra requires tool annotations that do not currently exist in the MCP specification. Our prompt-triggered composition observation (Section 6) is preliminary. The empirical evaluation (Section 7), while systematic, covers 17 servers and 15 compositions; broader coverage would strengthen generalisability claims. We also lack a formal ground-truth dataset for measuring precision and recall of composition surplus detection.

13. CONCLUSION

Multi-step tool chaining is a recognized and empirically validated attack vector [43][44]. Our contribution is to reframe this phenomenon through the lens of MCP/A2A governance: formalizing it as a capability-governance mismatch, parametrizing the composition operator by environment, policy, and objective, and demonstrating that the governance frameworks surrounding these protocols are structurally blind to compositional threats.

The empirical evaluation in Section 7 provides the first systematic evidence that composition surplus is a structural property of multi-server MCP deployments. Across 15 compositions drawn from 17 production servers, 100% produced emergent cross-server surpluses, with severity gradient as a reliable discriminator between high-risk and low-risk compositions. Triple compositions demonstrated superlinear growth in both attack surface and severity concentration.

As a preliminary finding, we report that a formally structured capability-class delta prompt framing appears to trigger composition behavior in frontier models via prompt injection, suggesting that injection and composition may constitute a single attack primitive. This observation requires rigorous validation—controlled experiments with ASR measurements, ablations, and comparison against STAC-style induction baselines.

Current governance frameworks—MCP authorization specifications, NIST standards, ISO/IEC 42001, OWASP guidance—evaluate tools in isolation. The composition surplus is invisible to per-tool security review, grows combinatorially with tool count, and is amplified by recursive MCP delegation. We have proposed a defense framework centered on capability algebra and composition-aware authorization, session-level capability tracking, delegation-depth bounds, and gateway-mediated policy enforcement, now validated by the scanner's empirical results.

The research agenda is urgent. The agentic AI community must develop: compositional threat modeling methodologies, capability vector annotations for the MCP specification, scalable composition analysis algorithms, rigorous evaluation of prompt-triggered composition, and governance frameworks that treat security as a system-level property rather than a per-component property. The 1+1=3 problem will intensify as models become more capable and tool ecosystems grow. Governance must learn to reason about compositions—because the models already do.

14. REFERENCES

[1] Anthropic, "Introducing the Model Context Protocol," Nov. 2024.
[2] A. Raina, "One Year of Model Context Protocol: From Experiment to Industry Standard," Dev.to, Nov. 2025.
[3] Google, "Announcing the Agent-to-Agent (A2A) Protocol," Apr. 2025.
[4] UK AI Security Institute, "Frontier AI Trends Report," 2025.
[5] N. Carlini et al., "The Attacker Moves Second," arXiv, Oct. 2025.
[6] Invariant Labs, "MCP Security Notification: Tool Poisoning Attacks," Apr. 2025.
[7] "Systematic Analysis of MCP Security," arXiv:2508.12538, Aug. 2025.
[8] Y. Wu et al., "Prompt Injection Attacks on Agentic Coding Assistants," arXiv:2601.17548, Jan. 2026.
[9] "Prompt Injection Attacks in Large Language Models and AI Agent Systems," MDPI Information, vol. 17, no. 1, Jan. 2026.
[10] "ToolHijacker: Prompt Injection Attack to Tool Selection in LLM Agents," in Proc. NDSS, Feb. 2026.
[11] A. Kharraz et al., "From Prompt Injections to Protocol Exploits," ScienceDirect, Dec. 2025.
[12] AuthZed, "A Timeline of Model Context Protocol (MCP) Security Breaches," 2025.
[13] E. Debenedetti et al., "Agent Security Bench (ASB)," in Proc. ICLR, 2025.
[14] NIST, "Announcing the AI Agent Standards Initiative," Feb. 2026.
[15] Docker, "MCP Security Issues Threatening AI Infrastructure," 2025.
[16] Practical DevSecOps, "MCP Security Vulnerabilities: Prevention Guide," Jan. 2026.
[17] JFrog Security Research, "CVE-2025-6514: Critical RCE in mcp-remote," 2025.
[18] "MCPTox: A Benchmark for Tool Poisoning Attack on Real-World MCP Servers," arXiv:2508.14925, Aug. 2025.
[19] Stack Overflow, "Is that allowed? Authentication and authorization in MCP," Jan. 2026.
[20] Model Context Protocol, "Specification 2025-06-18," Jun. 2025.
[21] Palo Alto Unit 42, "New Prompt Injection Attack Vectors Through MCP Sampling," 2025.
[22] Pillar Security, "The Security Risks of Model Context Protocol (MCP)," 2025.
[23] GitGuardian / Endor Labs, "Classic Vulnerabilities Meet AI Infrastructure," 2025.
[24] "Log-To-Leak: Prompt Injection Attacks on Tool-Using LLM Agents via MCP," OpenReview, 2025.
[25] Model Context Protocol, "Authorization — Specification 2025-03-26," Mar. 2025.
[26] A. Parecki, "Client Registration and Enterprise Management in the November 2025 MCP Authorization Spec," Nov. 2025.
[27] C. Posta, "The MCP Authorization Spec Is... a Mess for Enterprise," Nov. 2025.
[28] Docker, "MCP Security: Risks, Challenges, and How to Mitigate," 2025.
[29] Docker, "Dynamic MCPs: Stop Hardcoding Your Agents' World," 2025.
[30] ISO/IEC 42001:2023, "Artificial intelligence — Management system," 2023.
[31] Cloud Security Alliance, "AI Controls Matrix (AICM)," Jul. 2025.
[32] OWASP, "LLM01:2025 Prompt Injection," GenAI Security Project.
[33] Docker, "Docker MCP Gateway: Secure Infrastructure for Agentic AI," 2025.
[34] Auth0, "MCP Specs Update from June 2025: All About Auth," 2025.
[35] Microsoft, "Architecting Trust: A NIST-Based Security Governance Framework for AI Agents," 2025.
[36] Red Hat, "Model Context Protocol (MCP): Understanding Security Risks and Controls," 2025.
[37] CSET Georgetown, "AI Governance at the Frontier," Nov. 2025.
[38] Future of Life Institute, "2025 AI Safety Index," 2025.
[39] Z. Li et al., "Towards Verifiably Safe Tool Use for LLM Agents," in Proc. IEEE/ACM ICSE, 2026.
[40] A. Ranade et al., "Agents of Chaos: LLM Agent Failures," arXiv:2602.20021, Feb. 2026.
[41] Y. Chen et al., "Progent: Programmable Privilege Control for LLM Agents," arXiv, 2025.
[42] Z. Xu et al., "The Emerged Security and Privacy of LLM Agent: A Survey with Case Studies," ACM Computing Surveys, 2025.
[43] J.-J. Li et al., "STAC: When Innocent Tools Form Dangerous Chains to Jailbreak LLM Agents," arXiv:2509.25624, Sep. 2025.
[44] T. Jiang et al., "AgentLAB: Benchmarking LLM Agents against Long-Horizon Attacks," arXiv:2602.16901, Feb. 2026.
[45] "MCP-ITP: An Automated Framework for Implicit Tool Poisoning in MCP," arXiv:2601.07395, Jan. 2026.