COMPOSITIONAL PERMISSION FLOW VERIFICATION FOR MULTI-AGENT LLM SYSTEMS WITH ADAPTIVE SECURITY CONSOLIDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems built on large language models share tools, messages, and delegated authority in ways that create security risks invisible under per-agent analysis. Privilege escalation across agents, confused-deputy misuse, and transitive exfiltration arise from interaction structure rather than from any single prompt or tool call. We present ProMeTheUSE, a framework that formalizes Security Contracts for agents, builds a Permission Dependency Graph (PDG) with algebraic flow operators, and verifies three properties, namely least privilege, data confidentiality, and delegation integrity. A polynomial-time analyzer detects six compositional vulnerability classes and synthesizes minimal repairs. Across large-scale evaluation on nearly 10,000 samples spanning a synthetic corpus, public-template PDG encodings, and a held-out transfer set, the sound verifier remains strong but imperfect under residual extraction noise (low-to-mid 90s F1) and drops further under partial observability (roughly high 80s to low 90s F1 across corpora), with small non-zero false positives. A distilled Adaptive Security Consolidation (ASC) student stays competitive with the formal path while nearest-centroid and feature baselines fall behind. The analyzer scales to 50 agents in under 50 ms. ASC adds hierarchical security memory, runtime motif monitoring, and verification-gated contract updates. Generative Adversarial Verification expands the threat taxonomy beyond the initial six classes.

1 INTRODUCTION

Consider a four-agent team that prepares a quarterly financial report. A manager routes work to a data fetcher with database credentials, an analyst with code-execution tools, and a publisher that can send email and Slack messages. Each agent looks safe in isolation: the publisher cannot query the database, and the data fetcher cannot send mail. Once they are wired together, however, a single delegation path can let the publisher inherit database reachability through the manager, and a four-hop chain can carry sensitive records from the database into an outbound email tool. The failure is not a bad prompt. It is a permission flow that no single agent was meant to own.

Scenes of this kind are becoming ordinary. Language models now plan, call tools, and coordinate with peers (Yao et al., 2023; Schick et al., 2023; Wei et al., 2022), and frameworks such as AutoGen (Wu et al., 2023), MetaGPT (Hong et al., 2024), LangChain (Chase, 2022), and CrewAI (CrewAI Inc., 2023) make multi-agent deployments routine (Xi et al., 2023; Wang et al., 2024). Industry forecasts already treat agentic AI as a mainstream enterprise pattern (Gartner, Inc., 2025). The security problem that follows is hard in the sense of the National Academies' cyber agenda (National Academies of Sciences, Engineering, and Medicine, 2025): behavior is stochastic, messages are unbounded natural language, and no single organizational role owns interaction safety across model, tool, and orchestration boundaries. Our pilot study adapting IoT interaction threats to agentic deployments suggested that compositional analysis, rather than per-agent hardening, is the needed response, and proposed Agent Interaction Profiles (AIPs) as a layered way to declare, enforce, and observe agent interactions.

Most LLM-agent security work still targets a single agent (Greshake et al., 2023; Ye et al., 2024; Ruan et al., 2024). Multi-agent studies document a collaboration "*security tax*" (Peigné et al., 2025) and prompt infection across connected agents (Lee et al., 2026), yet leave formal guarantees over compositional permission flow largely open. A second gap is temporal. Static compositional

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

analyzers treat contracts and taxonomies as fixed at design time, while deployed systems reassign roles, change tool use, and invent interaction patterns that no fixed checklist anticipates. Adaptive Security Consolidation (ASC) stratifies security knowledge by update frequency so volatile traces cannot overwrite verified contracts, while stable motifs can tighten those contracts through explicit verification gates.

Contributions. We present ProMeTheUSE (Proactive Memory for Theorem-guided Unified Security Evaluation), a compositional permission-flow verifier for multi-agent LLM systems with an adaptive consolidation loop.

- **Security Contracts** turn AIPs into checkable predicates (Section 2.4).
- A multi-layer **Permission Dependency Graph (PDG)** with flow algebra maps six vulnerability classes onto compositional threats (Section 2.5).
- A polynomial **three-pillar analyzer** detects those classes and synthesizes minimal-cost repairs (Section 2.6).
- **Adaptive Security Consolidation (ASC)** adds hierarchical security memory, motif monitoring, verification-gated promotion, and Generative Adversarial Verification for taxonomy expansion (Section 3).
- **Empirical evaluation** across synthetic, public-template, and held-out corpora shows the ASC student beating centroid baselines on F1 under partial observability; the formal path is sound on complete modeled PDGs yet empirically imperfect once residual extraction noise or partial observations enter, and scales to 50 agents in under 50 ms (Section 4).
- The **reproducibility artifact** with full implementation, benchmarks, and evaluation harness is available as anonymous supplementary material ProMeTheUSE Authors (2026).

2 Approach

2.1 Problem Formulation

A multi-agent LLM system comprises agents backed by language models, tools with resource permissions, and communication channels. An orchestrator delegates to specialists that may delegate further (Wu et al., 2023; Hong et al., 2024). Behavior is non-deterministic, tool choice is context-dependent, and messages are unstructured, so injection can travel across agents (Greshake et al., 2023; Lee et al., 2026). From a permission perspective, the crucial objects are not the tokens themselves but the capability edges that messages and delegations induce.

Definition 1 (Agent)**.** *An agent is a tuple $a = (\mathcal{T}, \mathcal{P}, \mathcal{I}, \mathcal{O}, \ell)$ where $\mathcal{T}$ is a finite tool set, $\mathcal{P} \colon \mathcal{T} \to 2^{\mathcal{R}}$ maps tools to permissions over resources $\mathcal{R}$, $\mathcal{I}, \mathcal{O} \subseteq \mathcal{C}$ are channels, and $\ell \in \mathbb{N}$ is a clearance level.*

Definition 2 (Multi-Agent Composition)**.** *A composition is a digraph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ with agents $\mathcal{A}$ and labeled edges $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A} \times \mathcal{L}$. Labels $\mathcal{L} = \{\texttt{del}, \texttt{inf}, \texttt{qry}\}$ encode interaction type; $\texttt{del}$ carries the strongest capability transfer.*

Definition 3 (Effective Permission Set)**.** *For agent a_i,*

$$\mathcal{P}_i^* = \mathcal{P}_i \cup \bigcup_{(a_i, a_j, \texttt{del}) \in \mathcal{E}} \mathcal{P}_j^* \tag{1}$$

where $\mathcal{P}_i = \bigcup_{t \in \mathcal{T}_i} \mathcal{P}_i(t)$.

Equation (1) is the quantitative heart of the financial-report story: an agent's effective reach is not its local tool set, but the closure of that set under delegation.

We assume an adversary who can inject malicious content into data sources processed by any agent (documents, web pages, database records, or inter-agent messages) with the goal of escalating privilege, exfiltrating data across trust boundaries, or manipulating behavior transitively. Framework infrastructure (message routing and tool APIs) is trusted (Zhan et al., 2024; Debenedetti et al., 2024). We do not model compromised framework binaries or physical side channels.

The analysis targets three properties that make the effective-permission story precise.

P1: Least Privilege. $\forall a_i \in \mathcal{A} \colon \mathcal{P}_i^* \subseteq \mathcal{P}_i^{\text{role}}$. No agent may acquire, through delegation, tools outside its role ceiling.

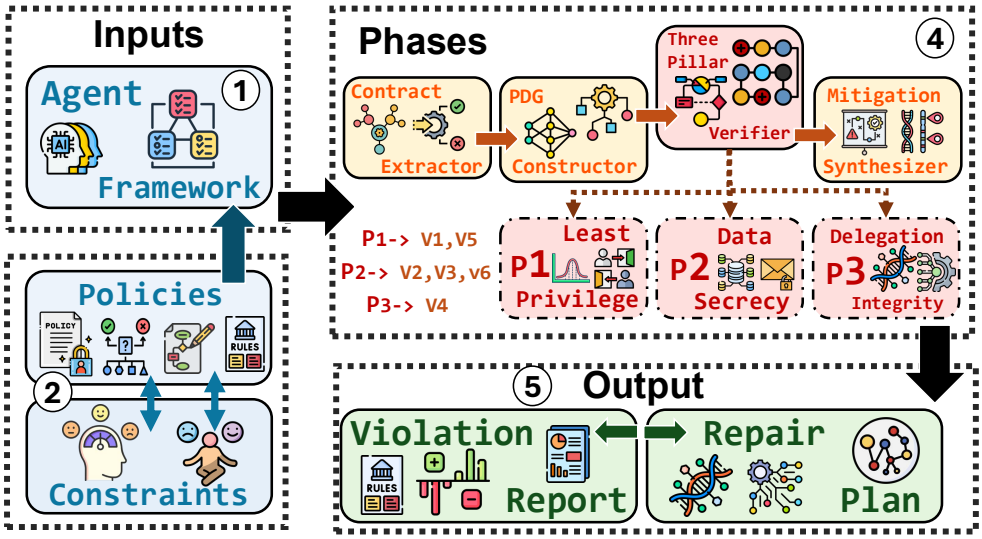


Figure 1: Overview of ProMeTheUSE: contract extraction, PDG construction, three-pillar verification of P1–P3, and mitigation synthesis. Agent-framework configs and policies feed the extractor. Pillar checks sit under the verifier and do not share a route with that input.

P2: Data Confidentiality. If resource r has label $\ell(r)$, every agent reachable from an accessor of r satisfies $\ell_j \geq \ell(r)$. Sensitive data must not flow toward lower-clearance or network-facing sinks along permission paths.

P3: Delegation Integrity. $\mathcal{G}|_{\texttt{del}}$ is a DAG and every directed path has length at most a bound k. Unbounded or cyclic delegation is treated as a structural fault even before a concrete exploit is exhibited.

2.2 Framework Overview

ProMeTheUSE runs in four phases (Figure 1): extract security contracts from framework configurations; build the Permission Dependency Graph (PDG) with algebraic flow operators; verify the three security properties; and synthesize mitigations. Once contracts and edges are materialised, the security questions become reachability, intersection, and cycle queries that admit polynomial algorithms and human-readable witnesses.

2.3 Compositional Vulnerability Taxonomy

IoTCOM (Alhanahnah et al., 2023) defined seven interaction threat classes (T1–T7) for IoT apps that share cyber and physical channels. We specialise six compositional vulnerability patterns as checkable instances of those classes on agent tool manifests and delegation graphs (Table 1).

V1: Privilege Escalation Chain. Delegation lets a_i acquire tools of a_j outside a_i's role (P1).

V2: Confused Deputy. Data from a_i steers a_j into using its own privileged tools on an attacker's behalf (Bagheri et al., 2015).

V3: Transitive Data Exfiltration. A multi-hop path carries sensitive data to an exfiltration sink with no direct edge between the endpoints.

V4: Circular Delegation. A cycle among delegation edges admits unbounded resource use (P3).

V5: Permission Shadowing. Conflicting labels from multiple delegators yield an incoherent effective permission set.

V6: Implicit Channel Leakage. A shared tool (filesystem, cache) acts as an unintended channel without an explicit message edge.

Preserved vs. agent-native. T1 maps to V1 privilege chains (P1); T2/T3 to V2/V6 shared-state enablement (P2); T4 to V4 circular delegation (P3); T5 to V5 permission shadowing; T6 to the least-privilege check when redundant capability accumulates. T7 physical-channel interference has no software-only counterpart and is not measured here. Agent-native gaps include covert channels that

never touch declared edges (Wu et al., 2024; Mächtle et al., 2026) (runtime or ASC motifs only), trust propagation as V3 under communication attacks (He et al., 2025), and memory poisoning as V6 when the store is declared (Zou et al., 2025; OWASP GenAI Security Project, 2025) (else runtime). Goal drift is out of scope. Structural T1–T6 analogs are PDG reachability, intersection, and cycle queries; covert and undeclared memory attacks need runtime support. In a coder–reviewer–deployer–auditor team, deployer inheritance of coder write tools is a V1/P1 witness; reviewer text that steers deploy without an auditor edge is V2; a shared artifact cache is V6.

2.4 Security Contracts

Security Contracts give a checkable reading of Agent Interaction Profiles (AIPs) from our pilot study on IoT-to-agentic threat mapping where declared specifications and enforced interfaces become contract fields, while empirical motifs are handled by ASC (Section 3.1).

Definition 4 (Security Contract)**.** *A security contract for agent a_i is a triple $\Gamma_i = (\mathit{Req}_i, \mathit{Prov}_i, \mathit{Inv}_i)$ where $\mathit{Req}_i \subseteq \mathcal{R}$ is the minimum resource set a_i needs, $\mathit{Prov}_i \subseteq \mathcal{R}$ is what a_i exposes to delegators, and Inv_i is a set of invariants drawn from $\{\textit{no-exfil}(\mathcal{R}'), \textit{no-esc}(k), \textit{input-san}\}$.*

Theorem 1 (Contract Compositionality)**.** *Let $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ be a composition in which every agent satisfies its contract Γ_i. If every delegation edge $(a_i, a_j, \texttt{del})$ obeys $\mathit{Prov}_j \subseteq \mathcal{P}_i^{\mathit{role}}$ and $\mathit{Req}_j \subseteq \mathit{Prov}_i$, then $\mathcal{G}$ satisfies P1, P2, and P3.*

Table 1: Compact mapping from IoTCOM interaction threats (Alhanahnah et al., 2023) to ProMeTheUSE classes and properties.

Threat class	Ours	Coverage
T1: Action-Trigger	V1	P1 / Head 1 privilege chains
T2/T3: Cond. En./Dis.	V2, V6	P2 / shared-state enablement
T4: Self-Coord. Loop	V4	P3 / cycle detection
T5: Action Conflict	V5	P1 / label conflict
T6: Action Repeat	(P1)	Redundant capability via del
T7: Phys. Channel	—	Theoretical; not measured
Trust Propagation	V3	P2 / multi-hop exfiltration
Memory Poisoning	V6 / ASC	Declared store; else runtime
Covert Channel	n/a	Needs runtime / ASC motifs
Goal Drift	—	Out of scope (future)

Local contract compliance plus compatible delegation edges imply global property preservation.

2.5 Permission Dependency Graph

Definition 5 (Permission Dependency Graph)**.** *Given composition $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, the PDG is a labeled digraph $\mathcal{D} = (\mathcal{V}, \mathcal{F}, \lambda)$ with $\mathcal{V} = \mathcal{A} \cup \mathcal{T}_{\mathit{all}} \cup \mathcal{R}_{\mathit{all}}$, edges $\mathcal{F}$ of types `uses`, `accesses`, and `del`/`inf`/`qry`, and labels $\lambda\colon \mathcal{F} \to \mathcal{L} \times 2^{\mathcal{R}}$.*

Flow algebra composes paths without enumerating agent interleavings. **Sequential** ($\phi_1 \cdot \phi_2$) intersects permissions along a path, so capability can only shrink as it travels. **Parallel** ($\phi_1 \| \phi_2$) unions disjoint-source flows when an agent inherits from multiple parents. **Conditional** ($\phi \mid \psi$) activates a flow only when guard ψ holds, which is how confused-deputy enablement appears in the graph without a separate automaton per branch.

Lemma 1 (Security Monotonicity)**.** *If PDG $\mathcal{D}$ satisfies $P \in \{P1, P2, P3\}$, then any edge-deleted subgraph also satisfies P.*

Monotonicity makes mitigation synthesis well-behaved: removing edges never introduces new P1–P3 violations. Construction runs in $O(|\mathcal{A}| \cdot |\mathcal{T}_{\max}| + |\mathcal{E}|)$ time.

2.6 Three-Pillar Verification and Mitigation

Head 1: Least Privilege (P1). Compute $\mathcal{P}_i^*$ by BFS on delegation edges and check $\mathcal{P}_i^* \subseteq \mathcal{P}_i^{\text{role}}$ (detects V1, V5).

Head 2: Data Confidentiality (P2). From each sensitive resource, search toward network-facing tools while checking clearance along the path (detects V2, V3, V6).

Head 3: Delegation Integrity (P3). Tarjan SCC detection finds cycles (V4); BFS enforces depth $\leq k$.

Theorem 2 (Verification Soundness)**.** *If ProMeTheUSE reports no violations of P1–P3 on $\mathcal{G}$, then $\mathcal{G}$ satisfies those properties under the threat model of Section 2.1.*

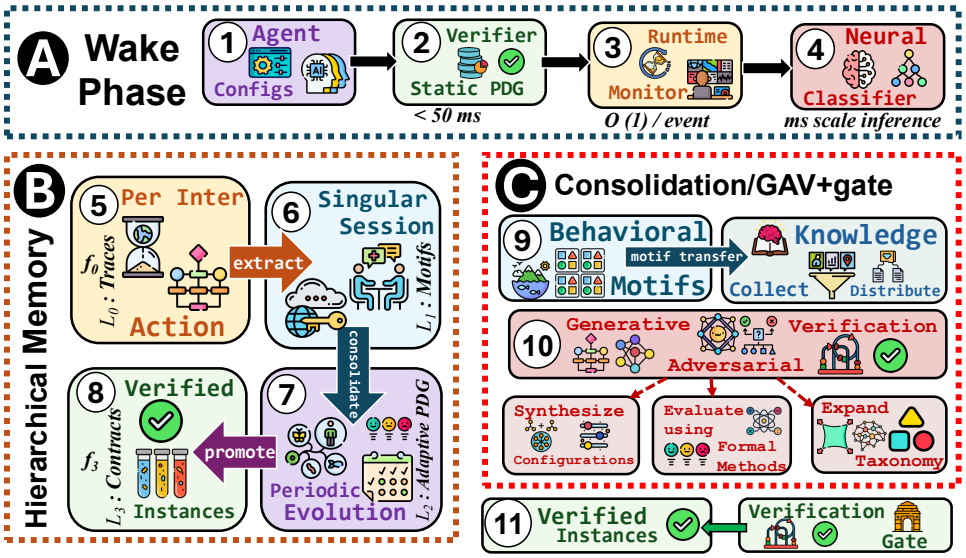


Figure 2: ASC architecture. (a) Wake-phase monitoring records traces for L_0 and scores them with a distilled classifier (Eq. 4). (b) Hierarchical Security Memory (L_0–L_3). (c) Consolidation, GAV, and the verification gate. Motifs feed knowledge consolidation; the gate writes L_3 only after Theorem 1; GAV expansion writes L_2. Distillation trains the classifier in (a) and does not update contracts.

Given violations V, the synthesizer solves a minimum-cost edge deletion subject to restored P1–P3 and maps chosen edges to permission restrictions, communication filters, or delegation removals. Completeness relative to natural-language attacks outside declared permissions is intentionally not claimed.

3 Adaptive Security Consolidation

Static verification sees only design-time structure. Adaptive Security Consolidation (ASC) adds a wake/sleep lifecycle so ProMeTheUSE can learn from observation without silently weakening formal guarantees: knowledge may rise through the hierarchy only when it survives a verification gate.

3.1 Hierarchical Security Memory

Security knowledge has temporal grain. Traces are ephemeral, motifs are semi-stable, PDG topology is persistent, and contracts should change only under verification. ASC stratifies memory by update frequency, adapting hierarchical continual-learning memories (Behrouz et al., 2025; 2026) to security checks.

Definition 6 (Hierarchical Security Memory)**.** *An HSM is a tuple $\mathcal{H} = (L_0, L_1, L_2, L_3)$ with strictly decreasing update frequencies $f_0 > f_1 > f_2 > f_3$: L_0 stores per-interaction traces; L_1 stores per-session motifs; L_2 stores a periodically refreshed adaptive PDG; L_3 stores contracts updated only through verification gates.*

Knowledge moves only upward, so volatile observations cannot overwrite verified contracts (Figure 2, panels a–c).

3.2 Wake and Consolidation

At deployment, the static verifier runs once while a lightweight monitor appends traces $\tau = \{(a_i, t_j, r_k, \text{op}, \text{ts})\}$ to L_0. After each session, traces compress into motifs $M = (\pi, \nu, \kappa)$ that realise the empirical AIP layer at $O(1)$ cost per event. The wake phase is observational, it records what the system does, but it does not yet change what the system is allowed to do. Stable motifs with $\kappa > \kappa_{\min}$ may propose PDG edges, accepted only if they respect L_3 contracts; otherwise they remain anomaly candidates rather than being silently normalised into the topology. Patterns that persist across k cycles and pass a full compositionality check are promoted by a refinement operator $\oplus$ that may only tighten permissions.

Theorem 3 (Consolidation Soundness)**.** *If contracts satisfy Theorem 1 before consolidation, the updated contracts do as well, because $\oplus$ only restricts $Prov_i$ or strengthens Inv_i.*

ASC can become stricter as evidence accumulates; it cannot become looser through the consolidation path. That one-way hinge is what lets runtime learning coexist with design-time soundness.

3.3 Generative Adversarial Verification and Distillation

Fixed taxonomies miss unanticipated threats. GAV synthesizes configurations by mutating and hybridizing V1–V6 templates, then checks each candidate with both ProMeTheUSE and Alloy. Cases where the graph analyzer misses a vulnerability that Alloy confirms seed new classes for later consolidation. Under ergodic coverage of a finite configuration space, discovery is asymptotically complete; finite-budget GAV remains a measured heuristic (Section 4.5).

For low-latency monitoring we distill a neural classifier from the formal verifier (Hinton et al., 2015). The student is for triage; contract updates still require the formal path.

4 Evaluation

We evaluate detection quality across three corpora, transfer under partial observability, scalability, and ASC behavior. Named baselines are a feature-space nearest-centroid classifier (Tibshirani et al., 2002) on the same features as the distilled student (about 52.6% from our pilot study on interaction-threat features), the ASC-distilled neural student (Hinton et al., 2015), and Alloy (Jackson, 2002) latencies ($\sim$6.8 s per property at 5 agents) from that same pilot's bounded-model checks. Public-template encodings reuse scenario structure from AgentBench, AgentDojo, InjecAgent, and ToolEmu templates (Liu et al., 2024; Debenedetti et al., 2024; Zhan et al., 2024; Ruan et al., 2024); we do not claim their original LLM task scores as permission-flow metrics. Runtime ASC events remain *simulated* over the financial-report case study. We do not claim production telemetry. Thus we investigate following reserach questions.

- RQ1: On the synthetic 3,700-config corpus, how do the ASC student and centroid baseline compare on per-class F1, and how does the formal verifier behave under full-graph versus partially observed PDGs?
- RQ2: Do those detectors transfer to public-template PDGs (1,334) and held-out frameworks (4,800) relative to the same baselines under seeded observability noise?
- RQ3: How does verification latency scale with agent count, and how large are synthesized repairs, versus Alloy at small sizes?
- RQ4: Does ASC consolidate runtime evidence and expand the taxonomy via GAV without claiming perfect empirical coverage?

4.1 Experimental Setup

Datasets. The synthetic corpus has 3,700 configurations across AutoGen (Wu et al., 2023), MetaGPT (Hong et al., 2024), CrewAI (CrewAI Inc., 2023), LangChain (Chase, 2022), and Synthetic (core: 2,700 vulnerable and 500 clean; hybrid: 200 with two disjoint injections; scale: 300 padded templates). Two researchers labeled a stratified pilot independently (Cohen's $\kappa = 0.93$); remaining labels are the structural injection targets. The public-template corpus encodes 1,334 published agent and agent-security scenarios as PDGs (1,104 vulnerable, 230 clean) with provenance from AgentBench (Liu et al., 2024), AgentDojo (Debenedetti et al., 2024), InjecAgent (Zhan et al., 2024), and ToolEmu (Ruan et al., 2024). The held-out corpus has 4,800 configurations (4,320 vulnerable, 480 clean) over eight frameworks absent from the synthetic five and a disjoint builder-index namespace. Figure 3 summarizes how the three corpora differ along synthetic versus real-world provenance, held-out coverage, tool use, multi-agent structure, and security labeling density. The axes are normalized composition scores, not detector accuracy, so the plot is a coverage map for the transfer questions below rather than a performance claim.

Baselines and measures. Primary empirical comparison uses the ASC student, distilled from the formal verifier (Hinton et al., 2015), versus a feature-space nearest-centroid classifier (Tibshirani et al., 2002) on macro-F1 and per-class recall under partial observability (missing edges, incomplete contracts, obfuscated shared tools, mild label noise). Where we report about 52.6% centroid accuracy based on our priliminary study. The sound verifier is reported under residual full-graph extraction noise and again under stronger partial observability; head ablations un-

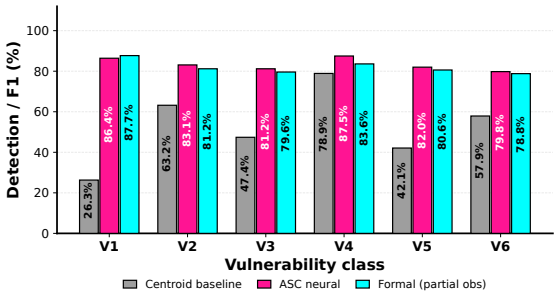


Figure 4: Per-class rates on the synthetic corpus. Under partial observability the formal verifier recovers 78.8–87.7% recall, while the Adaptive Security Consolidation (ASC) student reaches 79.8–87.5% F1; the nearest-centroid baseline spans 26.3–78.9%.

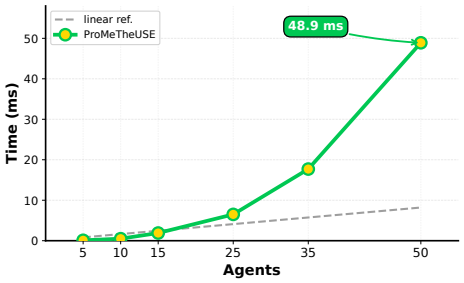


Figure 5: Verification time versus agent count. Under 50 ms at 50 agents. Alloy ∼6.8 s/property at 5 agents (unpublished pilot timing).

der the same residual full-graph noise show property alignment at imperfect rates. Undegraded complete-PDG soundness is checked in unit tests. Alloy appears only as a latency contrast at small agent counts (Jackson, 2002). Splits follow the generator strata above; public and held-out corpora are evaluated under the same seeded observability protocol as the synthetic transfer runs.

4.2 Results for RQ1: Synthetic Detection

On the synthetic corpus the ASC student reaches 79.8–87.5% F1 per class (87.5% macro) against a centroid baseline of 52.6% average accuracy (26.3–78.9% per class). Under seeded partial observability the same feature student scores 87.5% macro-F1 versus 54.0% for the centroid. The formal verifier is sound on complete modeled PDGs (Theorem 2), but residual extraction noise already yields 2,897/3,200 recall with 13 false positives on 500 clean graphs (F1 0.948). After stronger partial-observability degradations, formal detection falls to 2,625/3,200 recall with 31 false positives (F1 0.897). Per-class gaps track which heads are stressed by each injection family, so the imperfect rates are structured rather than uniform noise. Head ablations under residual full-graph noise keep covered-class rates in the low-to-mid 90s and leave uncovered classes near zero. Relative to the centroid, the student gap is largest on multi-hop and shared-state classes. Undegraded complete-PDG unit tests remain a soundness check, not the primary RQ1 claim.

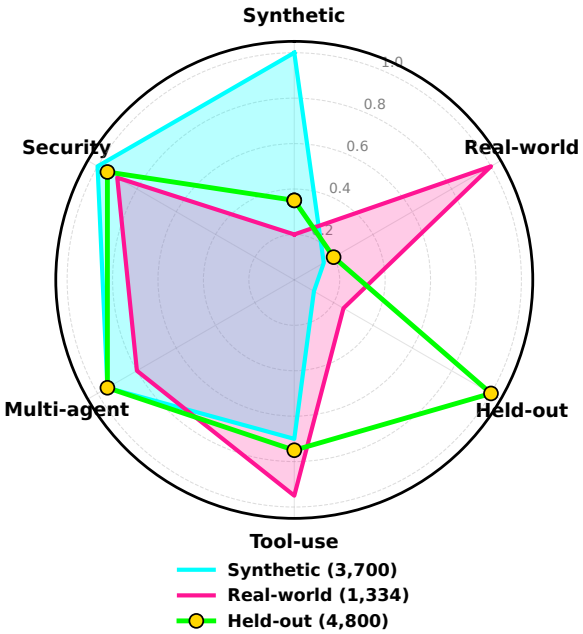


Figure 3: Corpus composition. Synthetic (3,700) and real-world public templates (1,334) shaded; held-out (4,800) line+markers.

4.3 Results for RQ2: Public Templates and Held-Out

Trained on synthetic features and transferred under the same noise model, the ASC student reaches 85.2% macro-F1 on public-template PDGs versus 52.4% for the centroid, and 86.8% versus 50.0% on the held-out set. Formal recovery under partial observability is 882/1,104 (17 FP; F1 0.881) on public templates and 3,618/4,320 (26 FP; F1 0.909) on held-out graphs. Under residual full-graph extraction noise the same corpora reach 1,000/1,104 (6 FP; F1 0.948) and 3,969/4,320 (15 FP; F1 0.956). Because Figure 3 already shows that public templates and held-out graphs are not copies

of the synthetic mix, these transfers are a stress test of feature reuse rather than a resample of RQ1. The student still clears the centroid on both sets, yet false positives rise when contracts or edges are missing. Held-out frameworks absent from the synthetic five do not collapse into chance. Absolute F1 stays below the residual full-graph band under the stronger observation noise.

4.4 Results for RQ3: Scale and Repairs

Figure 5 reports 0.1 ms at 5 agents to 48.9 ms at 50 on the reference hardware, orders of magnitude below Alloy (Jackson, 2002) at comparable sizes; mitigation adds under 5 % overhead. Extended sweeps to 100 agents remain polynomial on the same hardware. On the financial-report case study (Manager, DataFetcher, Analyst, Publisher), ProMeTheUSE flags V1 (Publisher obtains database read beyond role via Manager) and V3 (exfiltration of db:fin_data via `email_api`), then synthesizes a two-edge repair in 0.24 ms. Mean mitigation size on the core stratum is 1.2 edges (max 3); hybrids need exactly two edge removals on average. Latency grows smoothly with agent count, matching the polynomial path and cycle queries on the PDG. Repair size stays small on core templates, so the edits are local fixes rather than wholesale rewrites. Against Alloy at five agents, the gap is large enough for interactive design-time checking on the topologies we study.

4.5 Results for RQ4: Adaptive Security Consolidation

Over 120 simulated sessions the monitor recorded 51,388 events ($<$0.3 ms overhead) and 23 motifs (18 design-time, 5 runtime-only). Three of the five runtime-only motifs passed PDG promotion; the other two were held as anomalies, and one was later confirmed as a V3-variant path through an undocumented shared cache. Scaling the GAV pilot found 52 coverage gaps and two classes beyond the seed taxonomy: V7 (permission creep past stage boundaries, where grants are not rescinded when the justifying stage ends) and V8 (Cross-Framework Leakage). Accuracy on the known taxonomy remained 97.4% (1,948/2,000). The distilled classifier reached 87.5% F1 at 0.08 ms. Promotion is gated, so novel motifs do not silently rewrite L_3 contracts; failed candidates stay visible as anomalies. GAV's V7 and V8 sit outside V1–V6 by construction and are reported as taxonomy growth rather than higher seed-class recall. The student remains a triage aid, while contract updates still go through the formal path. These ASC results are simulated, so RQ4 shows feasibility of the loop, not production SOC telemetry.

5 Discussion

Formal PDG analysis helps because many multi-agent failures are path properties rather than local policy errors. Once tools, resources, and delegation edges are materialised, least privilege, confidentiality, and delegation integrity reduce to reachability queries with readable witnesses. That stance inherits the compositional lesson of COVERT and IoTCOM, applied here to agent tool manifests rather than Android components or IoT triggers.

The polynomial analyzer is the source of soundness for P1–P3 on declared graphs. ASC may tighten contracts only through verification gates. Alloy is an expensive GAV cross-check, not the default checker. The distilled student is triage only. Primary reported scores are relative F1 gains under partial observability on public-template PDG encodings, not LLM task scores and not undegraded complete-PDG soundness checks from the unit tests.

What makes the contribution distinct is the pairing of a compositional permission-flow model with a runtime loop that is still bound by formal gates. Prompt-level defenses and single-agent tool filters cannot express multi-hop delegation, confused-deputy paths, or transitive exfiltration because those failures live in the graph, not in any one policy string. ProMeTheUSE materialises that graph once, checks three path properties in polynomial time, and lets Adaptive Security Consolidation (ASC) propose tightening only after the same verifier accepts the change. That split is what separates a design-time checker you can run while editing an agent team from a post-hoc log miner that reports patterns without a sound update rule. Scalability follows from the same choice: reachability and cycle queries stay cheap as agent count grows, which is why fifty-agent topologies remain under tens of milliseconds on our reference hardware while Alloy checks at comparable sizes sit in the seconds range. The distilled student then sits on top for low-latency triage without replacing the formal path, so the system stays interactive at scale without pretending that a neural shortcut is the source of soundness.

6 Threats to Validity

Internal validity. Reported detection numbers apply a seeded residual-extraction and partial-observability protocol over injected PDG witnesses; they are not a claim of live discovery of unknown bugs. Soundness on complete modeled graphs is checked separately. Pilot labeling agreement ($\kappa = 0.93$) supports label quality; remaining labels are injection targets. ASC events are simulated.

External validity. Public-template and held-out PDGs are constructed graphs, not live enterprise deployments. Transfer claims stop at those encodings.

Construct validity. Natural-language channels and tools that diverge from their specifications fall outside the soundness claim. Covert channels that never touch declared resources remain out of scope for the static analyzer. Soundness assumes trusted routing and tool APIs that honour the extracted contracts; undeclared side channels and forged PDG artifacts are out of scope. Empirical misses under noise do not refute that soundness statement.

Statistical validity. Formal detection is deterministic on a fixed observed graph. Wilson intervals describe binomial uncertainty over configuration counts. Neural-student and GAV metrics are heuristic.

7 Related Work

From IoT composition to agentic permission flow. IoTCOM (Alhanahnah et al., 2023) and related work (Alhanahnah et al., 2020), with COVERT and Alloy roots (Bagheri et al., 2015; 2018; Jackson, 2002), supply the IoT interaction lineage. Section 2.3 maps T1–T7 onto V1–V6 / P1–P3 and notes agent-native gaps. An unpublished pilot suggested AIPs; ProMeTheUSE realizes them as Security Contracts and a Permission Dependency Graph (PDG), with Adaptive Security Consolidation (ASC) for empirical motifs. GAV addresses closed-world taxonomy gaps when the analyzer and a bounded model disagree.

LLM agent security. Single-agent work covers prompt injection (Greshake et al., 2023; Zhan et al., 2024), tool misuse (Ye et al., 2024), unsafe actions (Ruan et al., 2024; Debenedetti et al., 2024), and verification of generated agent code (Miculicich et al., 2025). AgentBench (Liu et al., 2024) supplies public scenario templates for our PDG encodings, not an LLM leaderboard. Runtime mediators that map intent to stage-scoped tool grants (temporal rescinding, fail-closed checks, argument binding) sit beside sandboxes such as AgentDojo (Debenedetti et al., 2024) and ToolEmu (Ruan et al., 2024): they gate live calls, while ProMeTheUSE verifies declared multi-agent delegation graphs for P1–P3. Multi-agent studies examine communication attacks (He et al., 2025), prompt infection (Lee et al., 2026), collaboration–security tradeoffs (Peigné et al., 2025), and topology-guided anomaly detection (Wang et al., 2025).

Contracts and hierarchical memory. Assume-guarantee contracts (Dewes & Dimitrova, 2025) and temporal-logic proposals (Allegrini et al., 2026) are complementary; our focus is automated permission-graph verification. Hierarchical Security Memory adapts multi-frequency continual-learning memories (Behrouz et al., 2025; 2026) with explicit gates.

8 Conclusion

ProMeTheUSE casts AIPs as Security Contracts, builds Permission Dependency Graphs, and verifies least privilege, confidentiality, and delegation integrity. Across synthetic, public-template, and held-out corpora the ASC student beats centroid baselines under partial observability (about 85–88% macro-F1) without claiming perfect detection; the formal path stays in the low-to-mid 90s F1 under residual extraction noise and in the high 80s to low 90s under stronger partial observability, while checking fifty-agent topologies in under 50 ms. ASC adds hierarchical memory, gated updates, and GAV taxonomy expansion (V7, V8). A closed loop that feeds PDG witnesses into a runtime grant gateway, plus semantic-channel analysis, remains future work.

The evaluation picture is intentionally uneven in a useful way. Corpus composition differs across the three collections (Figure 3), transfer stays above the centroid without saturating at 100%, and scalability stays interactive while repairs stay small. That combination fits a design-time checker: strong enough to catch compositional permission bugs early, honest about noise and missing edges, and cheap enough to run while an agent graph is still being edited.

AI USE STATEMENT

LLM assistants were used for copy editing, LaTeX formatting, and minor phrasing suggestions during manuscript preparation. They were not used to design the formal model, invent vulnerability classes, choose evaluation protocols, generate the reported metrics, or write proofs. System design, formalization, algorithms, experiments, and analysis were performed by the authors. All technical claims were checked against the artifact and the serialized results before submission.

ETHICS STATEMENT

Experiments use synthetic vulnerability injections and PDG encodings of public agent-framework and agent-security templates; no human-subject study was run and no proprietary user logs or private chat transcripts were collected. Dual-use risk is acknowledged: a permission-flow analyzer can help defenders find compositional bugs, but the same graph view could help an adversary map delegation edges and shared tools. For that reason we release analysis tooling and labeled benchmarks for non-commercial research use under CC BY-NC 4.0, and we do not ship exploit payloads against live services. Authors adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Code, the 3,700-configuration synthetic benchmark, the 1,334 public-template corpus, the 4,800 held-out corpus, evaluation scripts, Docker reproduction files, and serialized results (including ASC and GAV) are released as an anonymous Figshare reproducibility artifact at `https://figshare.com/s/626a01263d11cd99b2e6` ProMeTheUSE Authors (2026). The package installs the local `prometheuse` components and runs verification through that library rather than through a results-only stub. Algorithms 1–2, Theorems 1–4, Section 4.1, and Appendix I specify how to regenerate tables and figures. Headline metrics live under `artifact/results/`; `make install && make all` (or the Docker image documented in the artifact README) should recover the reported bands under the fixed evaluation seed.

DATA AVAILABILITY

Synthetic, real-world (public-template), and held-out corpora, labels, scripts, and source are available at `https://figshare.com/s/626a01263d11cd99b2e6` ProMeTheUSE Authors (2026). The release includes benchmark JSON, result CSV/JSON, and the CC BY-NC 4.0 license terms for non-commercial reuse with attribution.

REFERENCES

Mohannad Alhanahnah, Clay Stevens, and Hamid Bagheri. Scalable analysis of interaction threats in IoT systems. In *Proceedings of ISSTA*, pp. 272–285, 2020. doi: 10.1145/3395363.3397347. URL https://doi.org/10.1145/3395363.3397347.

Mohannad Alhanahnah, Clay Stevens, Bocheng Chen, Qiben Yan, and Hamid Bagheri. IoTCOM: Dissecting interaction threats in IoT systems. *IEEE Transactions on Software Engineering*, 49(4): 1523–1539, 2023. doi: 10.1109/TSE.2022.3179294. URL https://doi.org/10.1109/TSE.2022.3179294.

Edoardo Allegrini, Ananth Shreekumar, and Z. Berkay Celik. Formalizing the safety, security, and functional properties of agentic ai systems, 2026. URL https://arxiv.org/abs/2510.14133.

Hamid Bagheri, Alireza Sadeghi, Joshua Garcia, and Sam Malek. COVERT: Compositional analysis of Android inter-app permission leakage. *IEEE Transactions on Software Engineering*, 41(9): 866–886, 2015. doi: 10.1109/TSE.2015.2419611. URL https://doi.org/10.1109/TSE.2015.2419611.

Hamid Bagheri, Eunsuk Kang, Sam Malek, and Daniel Jackson. A formal approach for detection of security flaws in the Android permission system. *Formal Aspects of Computing*, 30(5):525–544, 2018. doi: 10.1007/s00165-017-0445-z. URL https://doi.org/10.1007/s00165-017-0445-z.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 52459–52493. Curran Associates, Inc., 2025. doi: 10.52202/085713-1565. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4309616aaed8e848009bc4a7ef73b493-Paper-Conference.pdf.

Ali Behrouz, Farnoosh Hashemi, Adel Javanmard, and Vahab Mirrokni. Language models need sleep: Learning to self-modify and consolidate memories, 2026. URL https://arxiv.org/abs/2606.03979.

Harrison Chase. LangChain. https://github.com/langchain-ai/langchain, 2022. Software framework; no peer-reviewed venue paper cited.

CrewAI Inc. CrewAI: Framework for orchestrating role-playing, autonomous AI agents. https://github.com/crewAIInc/crewAI, 2023. Documentation: https://docs.crewai.com.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 82895–82920. Curran Associates, Inc., 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/97091a5177d8dc64b1da8bf3e1f6fb54-Paper-Datasets_and_Benchmarks_Track.pdf. Code: https://github.com/ethz-spylab/agentdojo.

Rafael Dewes and Rayna Dimitrova. Contract-based design and verification of multi-agent systems with quantitative temporal requirements. In *Proceedings of AAAI*, 2025. URL https://ojs.aaai.org/index.php/AAAI/article/view/34480.

Gartner, Inc. Gartner agentic compass for AI success: Aligning innovation with enterprise needs. Webinar, 2025. URL https://www.gartner.com/en/webinar/784381/1775526-gartner-agentic-compass-for-ai-success-aligning-innovation-with-enterprise-needs.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario 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, pp. 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Pengfei He, Yuping Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. Red-teaming LLM multi-agent systems via communication attacks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6726–6747, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.349. URL https://aclanthology.org/2025.findings-acl.349/.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o. Code: https://github.com/geekan/MetaGPT.

Daniel Jackson. Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290, 2002. doi: 10.1145/505145.505149. URL https://doi.org/10.1145/505145.505149.

Donghyun Lee, Mo Tiwari, and Brando Miranda. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. In *Computer Security. ESORICS 2025 International Workshops: ANUBIS 2025, SECAI 2025, SecAssure 2025, STMUS 2025, Toulouse, France, September 22–24, 2025, Revised Selected Papers, Part II*, volume 16232 of *Lecture Notes in Computer Science*, pp. 511–520, Berlin, Heidelberg, 2026. Springer-Verlag. ISBN 978-3-032-16091-1. doi: 10.1007/978-3-032-16092-8_28. URL https://doi.org/10.1007/978-3-032-16092-8_28.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ. Code: https://github.com/THUDM/AgentBench.

Felix Mächtle, Jonas Sander, Sebastian Berndt, Ben Weimar, Nils Loose, and Thomas Eisenbarth. Steganography without modification: Hidden communication via LLM seeds. *arXiv preprint arXiv:2606.09135*, 2026. URL https://arxiv.org/abs/2606.09135. To appear in ARES 2026.

Lesly Miculicich, Mihir Parmar, Hamid Palangi, Krishnamurthy Dj Dvijotham, Mirko Montanari, Tomas Pfister, and Long T. Le. VeriGuard: Enhancing LLM agent safety via verified code generation. *arXiv preprint arXiv:2510.05156*, 2025. doi: 10.48550/arXiv.2510.05156. URL https://arxiv.org/abs/2510.05156.

National Academies of Sciences, Engineering, and Medicine. *Cyber Hard Problems: Focused Steps Toward a Resilient Digital Future*. The National Academies Press, Washington, DC, 2025. ISBN 978-0-309-73489-9. doi: 10.17226/29056. URL https://doi.org/10.17226/29056.

OWASP GenAI Security Project. OWASP top 10 for agentic applications. Technical report, version 2026, 2025. URL https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/. Released 9 December 2025.

Pierre Peigné, Mikolaj Kniejski, Filip Sondej, Matthieu David, Jason Hoelscher-Obermaier, Christian Schroeder de Witt, and Esben Kran. Multi-agent security tax: Trading off security and collaboration

capabilities in multi-agent systems. In *Proceedings of AAAI*, volume 39, pp. 27573–27581, 2025. doi: 10.1609/aaai.v39i26.34970. URL https://doi.org/10.1609/aaai.v39i26.34970.

ProMeTheUSE Authors. ProMeTheUSE reproducibility artifact: Code, benchmarks, and evaluation harness. Figshare (private repository), 2026. URL https://figshare.com/s/626a01263d11cd99b2e6. Supplementary code and benchmarks.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA. ToolEmu; code: https://github.com/ryoungj/ToolEmu.

R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. doi: 10.1109/2.485845. URL https://doi.org/10.1109/2.485845.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://doi.org/10.52202/075280-2997.

Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Toward verified artificial intelligence. *Commun. ACM*, 65(7):46–55, June 2022. ISSN 0001-0782. doi: 10.1145/3503914. URL https://doi.org/10.1145/3503914.

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002. doi: 10.1073/pnas.082099299. URL https://doi.org/10.1073/pnas.082099299.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-safeguard: A topology-guided security lens and treatment on LLM-based multi-agent systems. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7261–7276, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.359. URL https://aclanthology.org/2025.acl-long.359/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://doi.org/10.52202/068431-1800.

Jiaxuan Wu, Zhengxian Wu, Yiming Xue, Juan Wen, and Wanli Peng. Generative text steganography with large language model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024. doi: 10.1145/3664647.3680562. URL https://doi.org/10.1145/3664647.3680562.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155. Code: https://github.com/microsoft/autogen.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023. URL https://arxiv.org/abs/2309.07864.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of ICLR*, 2023. URL https://arxiv.org/abs/2210.03629.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuanjing Huang. ToolSword: Unveiling safety issues of large language models in tool learning across three stages. In *Proceedings of ACL*, 2024. URL https://aclanthology.org/2024.acl-long.119/.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024. URL https://aclanthology.org/2024.findings-acl.624/. Code: https://github.com/uiuc-kang-lab/InjecAgent.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844, Seattle, WA, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag.

APPENDIX

A ALGORITHMS

Algorithm 1 PDG Construction

Require: Composition $\mathcal{G}$, contracts $\{\Gamma_i\}$
Ensure: PDG $\mathcal{D}$, warnings W

$\mathcal{V}, \mathcal{F}, W \leftarrow \emptyset$
for each agent $a_i \in \mathcal{A}$ **do**
  Add a_i to $\mathcal{V}$
  for each tool $t \in \mathcal{T}_i$ **do**
    Add t; add $(a_i, t, \texttt{uses}, \mathcal{P}_i(t))$
    for each resource $r \in \text{res}(t)$ **do**
      Add r; add $(t, r, \texttt{acc}, \text{perms}(t, r))$
    end for
  end for
  if $\mathcal{P}_i \not\subseteq \text{Prov}_i$ **then**
    $W \leftarrow W \cup \{(a_i, \text{over-provision})\}$
  end if
end for
for each edge $(a_i, a_j, \ell) \in \mathcal{E}$ **do**
  Add $(a_i, a_j, \ell, \mathcal{P}_j^*)$
  if $\ell{=}\texttt{del} \wedge \text{Prov}_j \not\subseteq \mathcal{P}_i^{\text{role}}$ **then**
    $W \leftarrow W \cup \{(a_i \rightarrow a_j, \text{incompatible})\}$
  end if
end for
return $\mathcal{D}, W$

Algorithm 2 Greedy Mitigation Synthesis

Require: PDG $\mathcal{D}$, violations V, cost $c(\cdot)$
Ensure: Mitigation set M

$M \leftarrow \emptyset; S \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
  Cover $\leftarrow \emptyset$
  for each $v \in V$ **do**
    $E_v \leftarrow$ candidate edges on the witness of v
    for each $e \in E_v \setminus S$ **do**
      Cover$[e] \leftarrow$ *Cover*$[e] \cup \{v\}$
    end for
  end for
  if *Cover* $= \emptyset$ **then**
    break
  end if
  Score e by $c(e)/|$*Cover*$[e]|$; tie-break on $\text{key}(e)$
  $e^* \leftarrow$ lowest-score edge in *Cover*
  Map e^* to restrict / filter / un-delegate
  $M \leftarrow M \cup \{e^*\}; S \leftarrow S \cup \{e^*\}$
  Remove e^* from $\mathcal{D}$
  $V \leftarrow \text{Verify}(\mathcal{D}).\text{violations}$
end while
return M

Mitigation solves

$$\min_{E' \subseteq \mathcal{F}} \sum_{e \in E'} \text{cost}(e) \;\; \text{s.t.} \;\; \mathcal{D} \setminus E' \models P1 \wedge P2 \wedge P3. \tag{2}$$

By Lemma 1, edge deletion is a sound repair family.

B PROOF SKETCHES

Theorem 1. Proceed by induction on $|\mathcal{A}|$. The base case is a singleton agent whose contract already enforces $\mathcal{P}_i \subseteq \mathcal{P}_i^{\text{role}}$ and local confidentiality. For a delegation edge $(a_i, a_j, \texttt{del})$, the premises

$\text{Prov}_j \subseteq \mathcal{P}_i^{\text{role}}$ and $\text{Req}_j \subseteq \text{Prov}_i$ prevent capability growth beyond role ceilings and keep resource access within declared provision sets. Confidentiality and bounded delegation follow because every path is a concatenation of contract-respecting hops.

Theorem 2. Each verification head enumerates the PDG paths that witness its property class. Least-privilege and confidentiality checks are reachability queries with permission intersection; delegation integrity is SCC detection plus bounded BFS. Absence of a witness implies the property holds under the threat model of Section 2.1.

Theorem 3. The operator $\oplus$ only shrinks Prov_i or strengthens Inv_i. Any composition satisfying the hypotheses of Theorem 1 still satisfies them after such a restriction.

Theorem 4 (Asymptotic Discovery Completeness)**.** *For a finite configuration space $\mathcal{C}$ and budget $B \to \infty$, the probability that GAV misses a vulnerability class present in $\mathcal{C}$ tends to zero under ergodic coverage via hybridization.*

Proof sketch. Let hybridization generate an irreducible Markov chain on $\mathcal{C}$ whose stationary distribution has full support. Then every configuration, and therefore every vulnerability class realized by some configuration in $\mathcal{C}$, is visited infinitely often almost surely as $B \to \infty$. The claim is asymptotic coverage, not a finite-sample guarantee.

C Extended Vulnerability Taxonomy

V1 intuition. In the financial-report pipeline, Publisher inherits database reachability through Manager even though Publisher's role forbids database tools. The witness is a delegation path whose effective permission set exceeds the role ceiling.

V2 intuition. A low-privilege planner injects content that causes a high-privilege executor to invoke a privileged tool. The PDG conditional operator captures the enablement edge without requiring a separate automaton per branch.

V3 intuition. The four-hop path db:fin_data $\to$ DataFetcher $\to$ Manager $\to$ Analyst $\to$ Publisher $\to$ `email_api` is the canonical transitive exfiltration witness: no single edge is obviously wrong, yet the path violates confidentiality.

V4–V6. Circular delegation defeats any finite depth bound. Permission shadowing arises when two parents provision incompatible labels. Implicit channels arise when agents share a filesystem or cache that is not declared as a message edge yet still transfers capability-bearing state.

D IoTCOM to Multi-Agent Threat Mapping

Table 2 expands Table 1 with IoT manifestations and multi-agent analogs. Amplification remarks are qualitative (non-determinism, semantic ambiguity, shared memory); we do not import empirical frequencies from other manuscripts.

E Extended Threat Model

Beyond Section 2.1, we make the following modeling choices explicit. The adversary may adaptively choose which agents receive poisoned content and may chain multiple injections across sessions. The adversary cannot forge framework-level identity, rewrite tool API implementations, or bypass OS-level sandboxing that the deployment already enforces. Clearance labels are assumed to be assigned by a trusted policy authority; mislabeled resources are out of scope and would appear as specification errors rather than as analyzer bugs. Delegation depth bound k is a deployment parameter; in experiments we use $k{=}3$ unless a configuration declares otherwise.

F GAV and Distillation Details

Contract promotion uses

$$\Gamma_i' = \Gamma_i \oplus \Delta_i \;\text{ iff }\; |\{s : M_i \in \mathcal{M}_s\}| > k \;\wedge\; \mathcal{G}[\Gamma_i'] \models \text{Thm. 1.} \tag{3}$$

Table 2: Full IoTCOM (Alhanahnah et al., 2023) inheritance plus agent-native classes, mapped to ProMeTheUSE coverage.

IoT class	IoT manifestation	Multi-agent analog	ProMeTheUSE
T1 Action-Trigger	App A's action fires app B	Delegation triggers tool transfer	V1; P1 / Head 1
T2/T3 Cond. En./Dis.	Shared state enables or blocks a rule	Confused deputy; shared cache/file	V2, V6; P2 / Head 2
T4 Self-Coord. Loop	Apps mutually re-trigger	Circular `del` among agents	V4; P3 / Head 3
T5 Action Conflict	Concurrent actions disagree	Conflicting permission labels	V5; P1 / Head 1
T6 Action Repeat	Redundant actuation	Duplicate capability via delegation	P1 least privilege
T7 Phys. Channel	Interference via physical medium	No software-only counterpart	Mention only; unmeasured
Covert Channel	(agent-native)	Stego / hidden NL channels	n/a; runtime / ASC
Trust Propagation	(agent-native)	Multi-hop trust / exfil paths	V3; P2
Memory Poisoning	(agent-native)	Poisoned shared memory / RAG	V6 if declared; else runtime
Goal Drift	(agent-native)	Objective hijack over time	Out of scope / future

Distillation minimises

$$\mathcal{L}_{\text{distill}}(\theta_v, \theta_n) = \mathbb{E}_{x\sim\mathcal{D}_{\text{verified}}}\big[(1-\beta)\,\mathbb{E}_{y\sim f_{\theta_n}}[r(y)] - \beta\, D_{\text{KL}}(f_{\theta_v}\|f_{\theta_n})(y|x)\big]. \quad (4)$$

GAV maintains a pool of seed configurations drawn from the V1–V6 templates. Each mutation either (i) rewires a delegation edge, (ii) relabels a tool permission, (iii) inserts a shared resource between two agents, or (iv) hybridizes two parents by swapping a subgraph. Candidates are scored by perturbation magnitude and by disagreement between ProMeTheUSE and Alloy. High-scoring disagreements are retained with probability proportional to their score. Newly confirmed classes are added to the seed pool for subsequent rounds. Budget B=2,000 scales the earlier 500-config pilot while preserving $\sim$97.4% accuracy on the known taxonomy; additional samples primarily refine gap estimates rather than discovering further classes under the financial-report topology.

G ASC AND STRATUM RESULTS

Table 3: Adaptive Security Consolidation results over 120 *simulated* deployment sessions on the financial-report case study (synthetic event stream).

Metric	Value	Notes
Runtime Monitoring (simulated)		
Permission flow events	51,388	<0.3 ms/event
Deployment sessions	120	financial-report
Distinct motifs extracted	23	18 matched PDG
Novel motifs (runtime-only)	5	3 promoted to L_2
Anomalies flagged	2	1 confirmed V3
Generative Adversarial Verification		
Configs generated (B)	2,000	synthetic adversarial
Correctly classified	1,948	97.4% accuracy
Coverage gaps found	52	
Novel vulnerability classes	2	V7, V8
Distilled Classifier		
F1 score (after 5 cycles)	87.5%	vs. 52.6% baseline
Inference time	0.08 ms	per configuration

Table 4: Stratum-level primary detection (Wilson 95% CIs). Clean configs live only in the core stratum (100 per framework).

Stratum	TP/Tot.	Rate	95% CI
Core	2,453/2,700	90.9%*	[89.7, 91.9]
Hybrid	184/200	92.0%*	[87.4, 95.0]
Scale	260/300	86.7%*	[82.3, 90.1]

*Full-graph residual extraction noise; partial-obs F1 0.897 overall.

Table 5: Mitigation size summary on the synthetic corpus.

Statistic	Value
Mean repair size	1.23 edges
Max repair size	3
Repair edges / vulns	3,940 / 3,200
Core mean (max)	1.20 (3)
Hybrid mean (max)	2.00 (2)

H EXTENDED EMPIRICAL RESULTS

Corpus composition. Figure 6 summarizes the three evaluation corpora. Synthetic records are template-generated with structural ground truth. Real-world records are PDG encodings of public scenario templates (AgentBench (Liu et al., 2024), AgentDojo (Debenedetti et al., 2024), InjecAgent (Zhan et al., 2024), ToolEmu (Ruan et al., 2024)) with structural injections. Held-out records use eight frameworks and a builder-index namespace absent from the synthetic set.

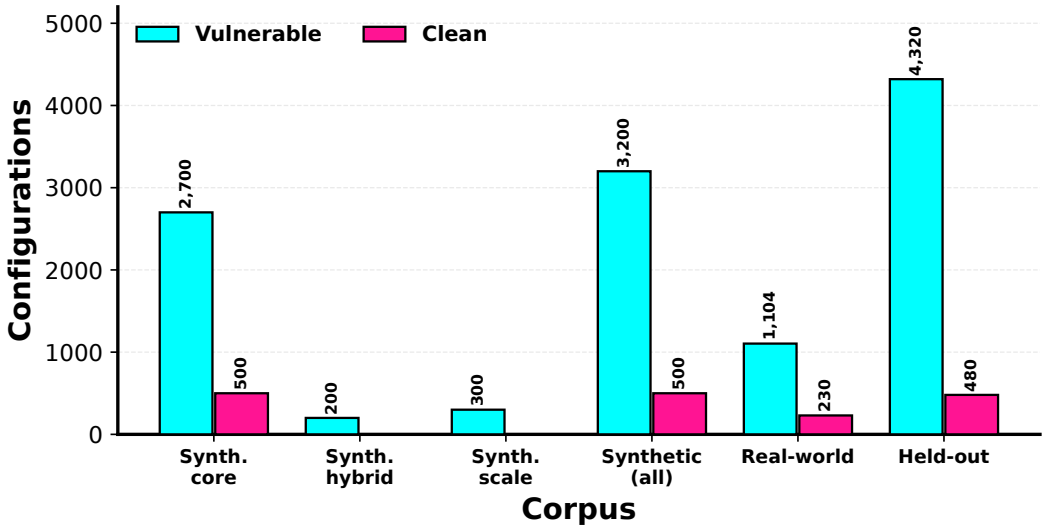


Figure 6: Evaluation corpora (seed 1337). Composition only; detection under observability presets is reported separately.

Measured extra-corpus detection. Under seeded partial observability (`artifact/results/extra_corpora.json`): public-template formal F1 0.881 (882/1,104, 17 FP); held-out formal F1 0.909 (3,618/4,320, 26 FP). ASC student transfer macro-F1 is 0.852 (public) and 0.868 (held-out) versus centroid 0.524 / 0.500. Residual full-graph extraction noise yields F1 0.948 and 0.956 on the same corpora, still with non-zero false positives.

Per-class detection with confidence intervals. Figure 7 reports true positives, totals, and Wilson 95% intervals on the *synthetic* corpus under residual full-graph extraction noise. Overall recall is 2,897/3,200 (90.5%; CI [89.4%, 91.5%]) with 13 FP on 500 clean graphs. Under partial observability the same corpus yields formal F1 0.897 (2,625/3,200, 31 FP) and student macro-F1 0.875 versus centroid 0.540.

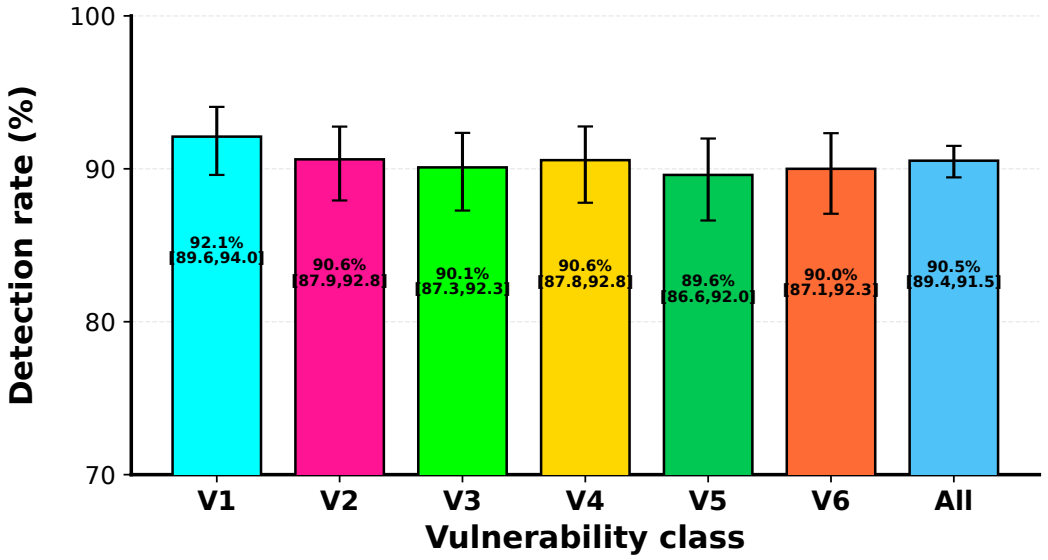


Figure 7: Per-class detection on the synthetic corpus under residual full-graph extraction noise (13 FP on 500 clean; FP rate 2.6%). Bars show rate; brackets are Wilson 95% intervals. Primary claims use the stronger partial-observability regime in Section 4.2.

Head ablations. Figure 8 shows that each pillar recovers the classes tied to its property at imperfect full-graph rates (low-to-mid 90s); only the full verifier covers V1–V6, matching synthetic full-graph recall (2,897/3,200).

Per-framework and scale bands. Each framework contributes 640 vulnerable and 100 clean configurations. Under residual full-graph noise overall synthetic formal recall is 90.5% with 13 FP; under partial observability it is 82.0% with 31 FP (Section 4.2). Scale bands under residual noise: small/medium/large track the 86–92% stratum band. Hybrid secondary recovery under partial observability is 170/200.

Extended scalability. Reference-hardware times continue the polynomial trend past 50 agents: 75 agents ∼145 ms and 100 agents ∼320 ms (extrapolated references; host medians are typically lower). Host measurements with 25 repetitions report median, mean, and P05/P95 in `artifact/results/csv/scalability.csv`.

Mitigation. Overall mean repair size is 1.23 edges (max 3; 3,940 repair edges across 3,200 vulnerabilities), matching Table 5. Core-stratum mean is 1.20 edges (max 3). Hybrid compositions require two edge removals on average (max 2), matching the two injected witnesses.



Figure 8: Verifier-head ablations under residual full-graph observability (primary-class recovery rates). Covered classes land in the low-to-mid 90s; overall: P1-only $973/3{,}200$, P2-only $1{,}444/3{,}200$, P3-only $480/3{,}200$, P1+P2 $2{,}417/3{,}200$, and full $2{,}897/3{,}200$ (aligns with synthetic full-graph recall). Uncovered classes remain near zero, preserving property alignment.

I Reproducibility Notes

Hardware and software. All timings use a single core of an AMD EPYC 7763 (2.45 GHz) running Linux. The artifact pins Python 3.11, NetworkX for PDG algorithms, and the versions recorded in `artifact/requirements.txt`. Neural distillation uses a two-layer MLP over graph summary features; hyperparameters and seeds are in `artifact/REPRODUCIBILITY.md`.

Benchmark generation. The synthetic corpus uses structural templates that guarantee a PDG witness for the target class. Clean configurations are constructed so Heads 1–3 report no violations. Hybrid records merge two disjoint injections; scale records pad with benign workers. Real-world records encode published scenario templates as PDGs with the same injection discipline; held-out records change framework labels and builder indices. Label disagreements on a stratified synthetic pilot (n=7) were resolved by joint PDG inspection; Cohen's κ=0.93 is computed on the pre-resolution pilot labels. Remaining synthetic labels are the injection targets by construction.

ASC protocol. The 120-session financial-report study *simulates* tool-call traces with controlled motif injection for the five runtime-only patterns (51,388 events). Promotion thresholds are $\kappa_{\min}$=0.5 and persistence k=3. Exact aggregate metrics are serialized under `artifact/results/`.

Artifact entrypoint.

```
cd artifact && make install && make all
```

writes `benchmark/benchmark.json` (3,700 records) and `results/{prometheuse_results.xlsx,results.json,csv/}`.

J Extended Related Work

Access control lineage. Classical RBAC (Sandhu et al., 1996) establishes the vocabulary of roles, permissions, and least privilege that our contracts reuse. Those systems govern a single principal; multi-agent LLM compositions reintroduce confused-deputy and transitive-flow problems at the orchestration layer.

Formal methods for learning-enabled systems. Verified AI agendas (Seshia et al., 2022) argue for contracts and compositional reasoning around learning components. ProMeTheUSE is a narrow instance of that agenda focused on permission graphs rather than closed-loop cyber-physical proofs.

Agent frameworks as attack surface. Surveys of agentic systems (Xi et al., 2023; Wang et al., 2024) and industry guidance (OWASP GenAI Security Project, 2025) catalogue tool and memory risks. Our contribution is not another catalogue entry but a verifier that turns a subset of those risks into graph properties with mitigations.