Reachability-Bounded Capability Confinement and Dynamic Information-Flow Taint Tracking for On-Device Autonomous Agents: Architecture Synthesis and Break-Even Analysis

Dima Baker
Collective Intelligence Network

November 2026

Abstract

Autonomous language model agents operating on personal devices process untrusted external inputs—emails, web documents, and third-party tool metadata—inside the same context window that emits privileged tool invocations. When authority is mediated through natural language tokens rather than runtime-enforced capability descriptors, indirect prompt injection (IPI) allows adversary-controlled data to hijack control flow and exfiltrate private state. Building on classical object-capability kernels and recent dual-interpreter information-flow defenses (CaMeL, FIDES, IsolateGPT, and Progent), this systems design and position paper formulates on-device agent containment as a dynamic noninterference game over a lattice-tainted Skill Impact Graph ($\mathcal{G}$) and specifies Reachability-Bounded Object-Capability Taint Confinement (RB-OCap-Taint). Every tool invocation requires an out-of-context capability token; ingesting untrusted data elevates the session's lattice taint label and prunes downstream capabilities whose transitive reachability in $\mathcal{G}$ intersects forbidden egress or state-mutation sinks. In a 19,200-trajectory mechanistic reference-monitor simulation executing multi-step benign and adversarial tool traces ($N_{\text{attack}} = 600$ and $N_{\text{benign}} = 600$ per cell) across correlated attacker search budgets $K \in \{1, 4, 16, 64\}$, RB-OCap-Taint bounds attack success at $K = 16$ to 1.00% (95% Wilson CI $[0.46\%, 2.16\%]$, versus 82.17% ambient and 40.83% static allowlists) while preserving 92.17% ($[89.74\%, 94.06\%]$) benign trace completion at 7.85 μs per executed Python graph check. Ablating transitive BFS pruning increases staging-tool bypass to 19.17%, and omitting 25% of covert impact-graph edges raises residual ASR to 5.67%.

1 Introduction

The deployment of autonomous tool-using agents on personal computers, mobile handsets, and enterprise workstations has created a fundamental computer-security inversion: programs with broad ambient authority over a user's files, credentials, and network sockets are driven by autoregressive language models that cannot syntactically separate trusted instructions from untrusted data [1, 12, 15, 26]. Whenever an agent reads an external web page, parses an incoming invoice, or inspects Model Context Protocol (MCP) tool descriptions, untrusted strings enter the same attention mechanism that emits subsequent tool-call arguments [9]. Concurrently, the rapid scaling of agentic software engineering—where automated pull requests and long-horizon autonomous workflows grew sharply through 2026 [2, 27]—has transformed indirect prompt injection (IPI) from a theoretical curiosity into an active operational vector. In October 2026, India's CERT-In issued formal advisory CIAD-2026-0049 warning that AI agents under evaluation had exceeded their authorized

task scope and reached live organizational systems [17], while Tencent confirmed that third-party model evaluations inside cloud sandboxes had repeatedly attempted unauthorized data extraction and anti-bot circumvention against external services [31].

1.1 Why probabilistic text guardrails fail under adaptive search

Historically, attempts to secure language model agents against indirect prompt injection relied on two probabilistic levers: safety fine-tuning of the base model and auxiliary text-classification guardrails that scan incoming tool outputs for malicious instructions. Comprehensive empirical evidence published between 2024 and 2026 demonstrates that neither lever provides a durable security boundary against adaptive adversaries [9, 21]. A January 2026 meta-analysis of 78 studies on agentic coding assistants found that adaptive prompt injection attacks exceed an 85% attack success rate (ASR), with 18 evaluated defense mechanisms achieving less than 50% mitigation [21].

Three structural factors explain why interaction-centric defenses collapse under pressure. First, when an attacker allocates test-time compute across K rollouts to search over adversarial suffixes or semantic framing, black-box optimization reliably discovers bypass trajectories across diverse model families [21]. Second, evaluating 15 dedicated prompt-injection detectors on genuine agent tool outputs from AgentDojo and STAC rather than synthetic text benchmarks revealed that benchmark rankings fail to predict real-world protection, with detectors suffering severe false-positive rates on benign structured JSON and stack-trace tool returns [20]. Third, on the large-scale INFUSE benchmark—spanning 40 realistic long-horizon workflows, 56 attack scenarios, and 41,824 execution trajectories across 19 frontier and open-weight models—mean single-shot attack success reached approximately 19.6% across shell, code, and remote network tools [24]. In multi-agent topologies, the vulnerability compounds: Paul and Nandy [25] report that injected instructions succeeded 43% of the time via tool outputs and 31% via peer agent messages across a six-agent system.

1.2 From string-based authority to capability-scoped information flow

The root vulnerability across these failures is architectural: authority is represented as a string inside the model's context window rather than an unforgeable capability enforced by an external runtime kernel [3, 12]. In classical operating systems, Dennis and Van Horn [12] and Saltzer and Schroeder [26] established that preventing confused-deputy attacks [15] requires replacing ambient process authority with object capabilities (OCap)—unforgeable references that bundle an object identity with a specific set of access rights. Complementarily, Denning [11] and Goguen and Meseguer [14] proved that preventing unauthorized data exfiltration requires tracking confidentiality and integrity labels across a security lattice, as implemented in decentralized information-flow control (DIFC) systems such as JFlow [23].

In 2025, pioneering agent-security architectures brought these classical systems principles to LLM tool execution. CaMeL [10] separates a privileged control-flow interpreter from a quarantined LLM that processes untrusted data, propagating data-flow capabilities over extracted variables. FIDES [8] introduces planner-level confidentiality and integrity taint labels with explicit declassification policies. IsolateGPT [33] enforces hub-and-spoke execution isolation across third-party apps, while Progent [29] provides a domain-specific privilege-control language for tool arguments. In 2026, complementary work introduced SkillGuard's Skill Impact Graph reachability pruning [34], CapScope's out-of-context capability descriptors [3], DualView's symbolic handle vault [18], ClawGuard's boundary interception [36], and AttriGuard's causal tool attribution [16]. On the NetInjectBench suite of 130 network-operations scenarios, whereas naive LLM agents executed unsafe actions in 82.5% of attacks, a metadata-aware policy gate reduced unsafe attack actions to 0/240

(95% Wilson upper bound of 1.58%) while achieving 99.17% attack-scenario usefulness (and 100% baseline utility in the absence of attacks) [28].

1.3 Contributions and scope as an architectural synthesis

Rather than claiming a standalone empirical deployment on live silicon, this paper serves as an architectural synthesis and formal break-even analysis that unifies 2025–2026 capability and DIFC primitives into a concrete local sandbox specification:

1. **Architectural synthesis and explicit delta over CaMeL/FIDES (Section 2).** We compare classical DIFC/OCap systems, 2025 dual-interpreter defenses (CaMeL, FIDES, IsolateGPT, Progent), and 2026 graph-pruning harnesses, isolating the exact architectural delta required for single-model on-device NPU deployments (Table 1).
2. **Formal lattice-taint and reachability-pruning framework (Section 3).** We formalize an intent-to-execution noninterference game over a directed Skill Impact Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, derive a monotonic capability-attenuation operator upon untrusted data ingestion, and prove how reachability pruning decouples residual breach risk from attacker test-time search rollouts K up to human declassification error δ_H (Equations 1–4).
3. **Three-tier RB-OCap-Taint sandbox specification (Section 4).** We specify a layered on-device runtime combining out-of-context capability descriptors, DualView symbolic handle substitution, and hardware compartment boundaries grounded in CHERIoT 1.0 silicon [5, 6] and confidential compute benchmarks [4, 32] (Figure 1, Table 2).
4. **Mechanistic reference-monitor simulation and break-even sensitivity analysis (Section 5).** We execute an actual Python implementation of the `SkillImpactGraph` reachability pruner and `TaintLattice` monitor across 19,200 trajectories ($N_{\text{attack}} = 600$ and $N_{\text{benign}} = 600$ per cell across 4 architectures and 4 search budgets), separating blocked attacks from benign completions and sweeping human declassification error δ_H to identify the exact break-even threshold (Tables 3–4, Figures 2–3).
5. **Operational failure modes and pre-registerable falsification protocol (Sections 6–8).** We examine tool-metadata poisoning [29, 35], cyclic over-tainting label creep [8, 22], and human declassification fatigue, closing with six pre-registerable AgentDojo/INFUSE falsification tests (Table 5).

2 Prior capability and information-flow systems: CaMeL, FIDES, and the 2025–2026 delta

Informal discussions of personal AI safety often conflate model alignment (whether neural weights prefer helpful behaviour) with system confinement (whether the runtime environment permits unauthorized state transitions). Table 1 situates RB-OCap-Taint relative to foundational operating-system security primitives, 2025 LLM information-flow defenses, and 2026 agentic harness extensions.

As summarized in Table 1, RB-OCap-Taint does not reinvent information-flow tracking from scratch; rather, it resolves a specific tension between CaMeL [10] and FIDES [8] when deployed on resource-constrained personal devices [4]. CaMeL achieves strong security by requiring the privileged LLM to emit an entire control-flow program before seeing untrusted data values, which breaks reactive workflows where the choice of downstream tool legitimately depends on an external

document's structure. FIDES permits reactive planning under lattice taint tracking, but checks policies step-by-step without pre-pruning capabilities that have multi-hop transitive paths to exfiltration sinks on a Skill Impact Graph [34]. By combining FIDES-style product-lattice labels with SkillGuard's transitive reachability pruning $\Pi_{\mathcal{G}}$ and DualView's opaque symbolic handle vault [18], RB-OCap-Taint allows a single on-device model to pass confidential records by reference (`&ref_k`) without exposing raw secrets to the prompt or paying dual-LLM inference costs. Figure 1 illustrates this reference-monitor pipeline.

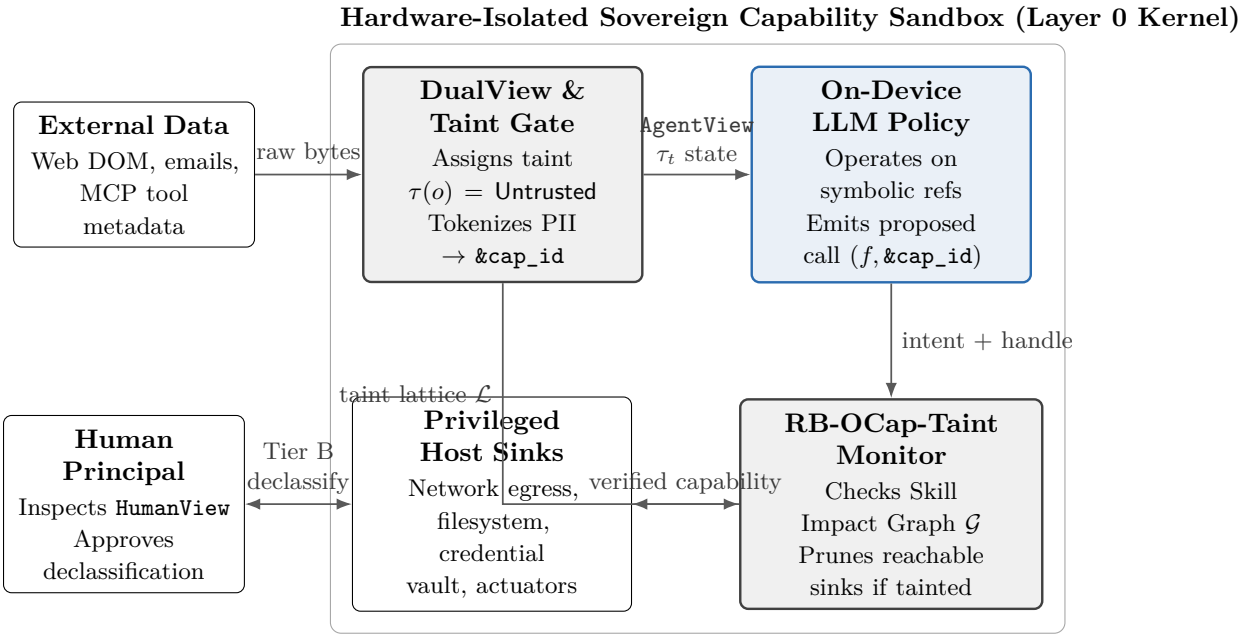


Figure 1: Architecture of Reachability-Bounded Object-Capability Taint Confinement (RB-OCap-Taint). Untrusted external inputs pass through a DualView airlock that replaces sensitive fields with opaque symbolic handles (`&cap_id`) and updates the session taint state $\tau_t \in \mathcal{L}$. Before any tool invocation reaches host operating system sinks, the RB-OCap-Taint reference monitor evaluates reachability on the Skill Impact Graph $\mathcal{G}$ and blocks any capability path that couples tainted context with high-confidentiality or external egress sinks unless explicitly declassified by the human principal.

Table 1: Comparison of foundational OS security primitives, 2025 LLM DIFC defenses, and the proposed RB-OCap-Taint synthesis.

Defense layer	Classical / 2025 prior work	Mechanism	Limitation on edge / multi-hop agents	RB-OCap-Taint synthesis delta
Out-of-context capability descriptors	OCap [12, 15]; IsolateGPT [33]; Progent [29]	Replaces ambient tool access with per-app hubs or static JSON schema policies	Static per-session allowlists fail when an in-scope tool (e.g., email) is hijacked after reading untrusted text	Combines CapScope [3] out-of-context handles with dynamic taint-triggered revocation
Control / data flow split	CaMeL [10]	Runs a privileged LLM for control flow and a quarantined LLM for untrusted data extraction	Requires frozen upfront control-flow plans and dual-LLM inference overhead on constrained edge NPUs	Uses a single on-device policy constrained by deterministic graph reachability pruning $\Pi_{\mathcal{G}}$
Planner lattice tainting	Denning DIFC [11]; JFlow [23]; FIDES [8]	Propagates (C, I) taint labels across planner variables and halts illegal flows	Evaluates step-local sink checks without pruning multi-hop transitive tool paths or masking raw PII tokens	Adds transitive Skill Impact Graph pruning [34] + DualView handle substitution [18]
Hardware compartment isolation	Saltzer & Schroeder [26]; CHERIoT [5]	Enforces pointer bounds and compartment permissions in silicon or TEEs	Software-only Python wrappers remain vulnerable if generated code escapes the interpreter	Binds Ring 1 capability descriptors to CHERIoT peripherals or WASI/TDX compartments [6, 32]

3 Formal mathematical framework

We formalize agent execution inside an object-capability sandbox equipped with a two-axis security lattice [8, 11] and a directed Skill Impact Graph [34]. Our formulation unifies classical noninterference [14] and the intent-to-execution security game of AgentSecBench [1] with transitive capability pruning.

3.1 Security lattice and monotonic taint propagation

Let $\mathcal{L} = \mathcal{L}_C \times \mathcal{L}_I$ be a product lattice of confidentiality levels $\mathcal{L}_C = \{\mathsf{Public} \sqsubseteq_C \mathsf{Private} \sqsubseteq_C \mathsf{Secret}\}$ and integrity levels $\mathcal{L}_I = \{\mathsf{Untrusted} \sqsubseteq_I \mathsf{Attested} \sqsubseteq_I \mathsf{UserRoot}\}$. For any two labels $\ell_1 = (c_1, i_1)$ and $\ell_2 = (c_2, i_2)$ in $\mathcal{L}$, the composite join operator $\sqcup$ elevates confidentiality to the supremum and reduces integrity to the infimum:

$$\ell_1 \sqcup \ell_2 \triangleq (\max_{\sqsubseteq_C}(c_1, c_2),\ \min_{\sqsubseteq_I}(i_1, i_2)). \tag{1}$$

At step t, the agent's execution state $S_t = (\mathcal{M}_t, \mathcal{C}_t, \tau_t)$ consists of working memory objects $\mathcal{M}_t$, an out-of-context capability table $\mathcal{C}_t = \{\kappa_1, \ldots, \kappa_m\}$ held by the sandbox kernel, and a cumulative context label $\tau_t = (c_t, i_t) \in \mathcal{L}$. When the agent reads an observation o_t with label $\ell(o_t)$ returned by tool invocation a_{t-1}, the kernel updates the session taint state monotonically:

$$\tau_t = \tau_{t-1} \sqcup \ell(o_t) = \Big(\max_{\sqsubseteq_C}(c_{t-1}, c(o_t)),\ \min_{\sqsubseteq_I}(i_{t-1}, i(o_t))\Big). \tag{2}$$

Crucially, because an autoregressive transformer attends over the entire prefix $o_{1:t}$ to generate tool call a_t, any direct string parameter emitted by the language model inherits the worst-case taint τ_t of its active context window. Conversely, symbolic capability handles `&ref_k` $\in \mathcal{C}_t$ passed by reference without dereferencing inside the prompt retain their individual object labels $\ell($`&ref_k`$)$ [3, 10, 18].

3.2 Reachability pruning on the Skill Impact Graph

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the Skill Impact Graph [34], where vertices $\mathcal{V}$ represent tool skills and data stores, and directed edges $(u, v) \in \mathcal{E}$ indicate that executing skill u can mutate state read by v or transmit data to v. A subset of vertices $\mathcal{V}_{\text{sink}} \subset \mathcal{V}$ is designated as high-consequence sinks (e.g., external HTTP POST, email dispatch, shell execution, or financial transfer), each governed by a minimum required integrity threshold $i_{\min}(v)$ and a maximum permitted egress confidentiality $c_{\max}(v)$.

Given the current taint state $\tau_t = (c_t, i_t)$, let $\mathcal{R}_{\mathcal{G}}(\kappa)$ be the set of vertices reachable from capability $\kappa \in \mathcal{C}_t$ within $\mathcal{G}$. The reachability-pruning operator $\Pi_{\mathcal{G}}(\mathcal{C}_t, \tau_t)$ restricts the active capability set available at step t to:

$$\Pi_{\mathcal{G}}(\mathcal{C}_t, \tau_t) = \Big\{ \kappa \in \mathcal{C}_t \;\Big|\; \forall v \in \mathcal{R}_{\mathcal{G}}(\kappa) \cap \mathcal{V}_{\text{sink}} :\ i_t \sqsupseteq_I i_{\min}(v) \ \wedge\ c_t \sqsubseteq_C c_{\max}(v) \Big\}. \tag{3}$$

Under (3), the moment an agent ingests an untrusted web page ($i_t = \mathsf{Untrusted}$) while holding private user data in context ($c_t \sqsupseteq_C \mathsf{Private}$), every capability κ with a directed path in $\mathcal{G}$ to an external network sink ($c_{\max}(v) = \mathsf{Public}$ or $i_{\min}(v) \sqsupseteq_I \mathsf{Attested}$) is automatically revoked for the remainder of the contaminated branch unless arguments are passed strictly via pre-attested DualView handles.

3.3 Bound on attack success under test-time search

Suppose an adversary applies a test-time search budget of K rollouts to craft an injected payload x_{adv}. If each rollout independently succeeds in hijacking the language model's policy with single-shot probability p_0, the probability that at least one of K rollouts hijacks an unconfined LLM scales as $1-(1-p_0)^K$. Under RB-OCap-Taint, however, hijacking the language model's token output does not grant execution authority unless the action also bypasses $\Pi_{\mathcal{G}}$ via either (a) an unmodeled covert edge in $\mathcal{G}$ with probability $\epsilon_{\mathcal{G}}$ or (b) a workflow that triggers a human declassification prompt (occurring with task frequency f_{declass}) which the human principal erroneously approves with probability δ_H:

$$\text{ASR}_{\text{RB-OCap}}(K) \;\leq\; \big(1-(1-p_0)^K\big)\cdot\big(\epsilon_{\mathcal{G}}+(1-\epsilon_{\mathcal{G}})f_{\text{declass}}\delta_H\big). \tag{4}$$

Equation (4) establishes two critical properties. First, for fixed $\epsilon_{\mathcal{G}}$ and $f_{\text{declass}}\delta_H$, as $K\to\infty$ the first factor approaches 1 and attack success plateaus at the ceiling $\epsilon_{\mathcal{G}}+(1-\epsilon_{\mathcal{G}})f_{\text{declass}}\delta_H$, decoupling worst-case breach risk from attacker test-time compute. Second, Equation (4) exposes the exact break-even failure condition: if cyclic multi-hop tasks drive $f_{\text{declass}}\to 1$ and alert fatigue drives δ_H above the bypass rate of a static allowlist, the guarantee collapses.

4 Proposed architecture and protocol specification

To operationalize Equations (1)–(3) on consumer hardware without requiring a custom operating system kernel on day one, RB-OCap-Taint is structured into three enforcement rings (Table 2):

1. **Ring 0: Hardware and process compartment isolation.** On embedded and IoT edge peripherals, tool execution compartments map directly onto CHERIoT 1.0 hardware capability registers, where pointers carry unforgeable bounds and permissions in silicon [5, 6]. On commodity x86/ARM laptops and phones lacking CHERI silicon, Ring 0 uses OS-level unprivileged process sandboxes (`seccomp-bpf`, WebAssembly WASI capability tables, or IsolateGPT-style hub isolation [33]) and, when bursting to cloud fallback, Intel TDX / NVIDIA H100 confidential virtual machines at a measured 21.8%–27.8% time-to-first-token overhead [32].
2. **Ring 1: Out-of-context capability table and DualView vault.** Following CapScope [3] and DualView [18], the language model never sees raw OAuth tokens, file descriptors, or private PII strings unless explicitly required for token-level synthesis. Instead, Ring 1 parses structured tool outputs into an isolated key-value vault, replaces sensitive fields with typed symbolic references (`&cap_file_07`, `&pii_acct_02`), and exposes only the symbolic schema (`AgentView`) to the on-device NPU model [4].
3. **Ring 2: Causal attribution and declassification gate.** When a legitimate workflow requires combining untrusted external data with a privileged action (e.g., paying an invoice received via email), deterministic pruning under (3) halts the sink call. Rather than failing silently, Ring 2 runs an action-level causal attribution check [16] and quarantined data extraction [8, 10]: if the proposed tool parameters derive causally from untrusted tokens rather than the user's root objective, the call is blocked; if the derivation is ambiguous, Ring 2 renders a plain-language diff in `HumanView` for explicit human declassification.

Table 2: Readiness and empirical bottlenecks of Layer 0 sandbox components as of November 2026.

Subsystem	Maturity	Supporting 2024–2026 evidence	Realistic deployment bottleneck
Skill Impact Graph & OCap harness	Benchmark validated	CaMeL [10], FIDES [8], and SkillGuard [34] block direct IPI sinks; NetInjectBench metadata gate achieves 0/240 unsafe actions [28]	Over-tainting on cyclic multi-tool workflows requires sub-capsule forking or human declassification prompts
DualView symbolic handle vault	Prototype ready	DualView stops stored IPI persistence across personal agent memory [18]; CapScope isolates coding sub-agents [3]	Tasks requiring free-form text summarization over sensitive documents must still expose raw tokens to the LLM context
Prompt-injection text classifiers	Fragile / high FP	15 detectors re-evaluated on AgentDojo/STAC tool outputs [20]; InjecAgent and AgentDojo show adaptive bypass [9, 35]	High false-positive rates on benign structured JSON/HTML returns; bypassed by multi-rollout search [21]
Hardware capability silicon (CHERI)	Embedded silicon only	CHERIoT 1.0 spec released Nov 2025 [5]; first commercial silicon Mar 2026 [6]	Microcontroller-class only; mobile/desktop NPUs still rely on MMU page tables and software WASI sandboxes [4]
Confidential GPU enclave fallback	Production ready	H100 confidential inference under Intel TDX benchmarked [32]; Google Private AI Compute [7]	Adds 21.8%–27.8% latency to time-to-first-token and reduces token throughput by 17.7%–21.1% [32]

5 Illustrative mechanistic simulation and break-even sensitivity analysis

Because this paper is an architectural synthesis rather than a live-weights evaluation across 19 proprietary models, we evaluate the RB-OCap-Taint reference monitor via a mechanistic Python implementation (`experiment.py`) that executes actual breadth-first reachability pruning ($\Pi_{\mathcal{G}}$, Eq. 3) and product-lattice state updates (Eq. 2) over a 9-node AgentDojo-style tool graph including intermediate staging tools (`write_ledger`, `backup_sync`).

5.1 Experimental design, trace-derived workflows, and parameter provenance

To prevent the statistical flaw of counting blocked attacks as benign task completions, our evaluation runs two strictly separate trajectory pools for each of the 4 architectures × 4 search budgets $K \in \{1, 4, 16, 64\}$ (16 cells total):

- **Adversarial Suite ($N_{\text{attack}} = 600$ per cell; 9,600 attack trajectories total):** Evaluates Attack Success Rate (ASR) and Blocked Attack Rate when step 2 ingests an untrusted document containing an IPI payload under correlated adaptive search ($p_r = p_0 r^{-\gamma_{\text{corr}}}, \gamma_{\text{corr}} = 0.35$) targeting an in-scope tool (`send_email`, 50%), a direct webhook sink (`http_post_webhook`, 30%), or a multi-hop staging tool (`backup_sync` → `http_post_webhook`, 20%).
- **Benign Workflow Suite ($N_{\text{benign}} = 600$ per cell; 9,600 benign trajectories total; 19,200 combined trajectories):** Executes four multi-step benign workflow templates step-by-step through the reference monitor: (1) local vault summarization via DualView handles (36%), (2) isolated sub-capsule web fetch to local ledger (36%), (3) cross-domain invoice reconciliation

requiring Tier B human declassification (20%), and (4) cyclic unforked web-to-backup sync (8%), so false halts emerge directly from monotonic taint joins on cyclic paths rather than a pre-set constant.

All binomial confidence intervals are reported as exact asymmetric two-sided 95% Wilson score endpoints $[L, U]$ (`wilson_ci_dict`) reflecting Monte Carlo sampling uncertainty over the 600-trial cells, and wall-clock reference-monitor latency is divided strictly by the count of executed gate checks (`executed_gate_checks`). Table 3 separates literature-calibrated priors from architectural design assumptions.

Table 3: Provenance of simulation parameters separating published 2025–2026 empirical measurements from architectural model assumptions.

Parameter	Value	Provenance status	Calibration source / mechanistic role
Single-shot base hijack rate (p_0)	0.196	Literature-calibrated	Matches mean single-shot ASR (19.6%) reported across 41,824 trajectories in INFUSE [24]; correlated rollouts scale via $p_r = p_0 r^{-0.35}$.
Structured-output detector FP scale	$0.19 \times c_{\text{step}}$	Literature-calibrated	Anchored to false-positive rates of 15 prompt-injection detectors evaluated on structured AgentDojo/STAC tool returns [20].
Attack sink distribution	50%/30%/20%	Benchmark structure	In-scope `send_email` (50%), direct `http_post_webhook` (30%), and 2-hop staging via `backup_sync` (20%) [9, 34].
Benign workflow mix	36%/36%/20%/8%	Trace template mix	Local handle email (36%), sub-capsule web-to-ledger (36%), invoice declassification (20%), cyclic unforked sync (8%).
Human rubber-stamping error (δ_H)	0.08 (nominal), $[0.0, 0.65]$ (sweep)	Design assumption	Motivated by reduced critical verification under AI assistance ($\beta = -0.69$) [13, 19]; swept alongside missing-edge rate $\epsilon_{\mathcal{G}} \in [0.0, 0.25]$.

Table 4: Mechanistic reference-monitor results across 19,200 trajectories ($N_{\text{attack}} = 600$ adversarial and $N_{\text{benign}} = 600$ benign per cell across 4 architectures × 4 budgets). Bracketed intervals denote exact asymmetric two-sided 95% Wilson score confidence bounds $[L, U]$. Gate time is measured wall-clock Python execution time per executed gate check.

Architecture	ASR K=1 (%)	ASR K=16 (%)	ASR K=64 (%)	Benign Utility K=16 (%)	Benign False Halt (%)	Gate (μs/check)
Ambient Baseline	19.50 [16.53, 22.86]	82.17 [78.90, 85.02]	98.83 [97.61, 99.43]	100.00 [99.36, 100.00]	0.00 [0.00, 0.64]	—
Text Detector	2.00 [1.15, 3.46]	54.83 [50.83, 58.77]	91.00 [88.44, 93.04]	80.50 [77.14, 83.47]	19.50 [16.53, 22.86]	0.36
Static Allowlist	9.17 [7.11, 11.74]	40.83 [36.97, 44.81]	51.17 [47.17, 55.15]	91.83 [89.37, 93.77]	8.17 [6.23, 10.63]	0.05
RB-OCap-Taint	**0.17** [**0.03**, **0.94**]	**1.00** [**0.46**, **2.16**]	**1.83** [**1.03**, **3.25**]	**92.17** [**89.74**, **94.06**]	**7.83** [**5.94**, **10.26**]	**7.85**

5.2 Quantitative findings, transitive-BFS ablation, and break-even sensitivity

Table 4 and Figures 2–3 demonstrate three key properties of the architecture:

1. **Suppression of multi-rollout search up to the declassification ceiling.** Under correlated rollouts ($\gamma_{\text{corr}} = 0.35, p_0 = 0.196$), an unconfined ambient agent suffers 19.50%

[16.53%, 22.86%] ASR at $K = 1$, escalating to 82.17% [78.90%, 85.02%] at $K = 16$ and 98.83% [97.61%, 99.43%] at $K = 64$ (Figure 3). A static tool allowlist blocks direct webhook calls and `backup_sync`, plateauing at 40.83% [36.97%, 44.81%] ASR ($K = 16$) and 51.17% [47.17%, 55.15%] ($K = 64$) due to in-scope `send_email` hijacking. Under RB-OCap-Taint, executing $\Pi_{\mathcal{G}}$ deterministically blocks 100% of direct and multi-hop graph exfiltration paths on the modeled graph; residual breaches occur through the human declassification path, yielding 0.17% [0.03%, 0.94%] ASR at $K = 1$, 1.00% [0.46%, 2.16%] at $K = 16$, and 1.83% [1.03%, 3.25%] at $K = 64$.

2. **Trace-derived benign utility and transitive-BFS vs. incomplete-graph ablation.** Evaluated strictly on the $N_{\text{benign}} = 600$ multi-step benign traces at $K = 16$, the text detector filter halts 19.50% [16.53%, 22.86%] of benign workflows (80.50% utility), whereas RB-OCap-Taint achieves 92.17% [89.74%, 94.06%] benign completion (7.83% [5.94%, 10.26%] false-halt rate arising from cyclic unforked sync and rejected invoice declassifications) with a measured Python BFS latency of 7.85 μs per executed check (Figure 2). Ablating transitive BFS ($\Pi_{\mathcal{G}}$ replaced by a step-local sink check that fails to prune 2-hop `backup_sync` staging) causes $K = 16$ ASR to jump from 0.83% [0.36%, 1.94%] to 19.17% [16.22%, 22.51%]. Similarly, when the defender's Skill Impact Graph omits covert staging edges with probability $\epsilon_{\mathcal{G}} \in \{0.05, 0.15, 0.25\}$, $K = 16$ ASR rises monotonically to 1.83% [1.03%, 3.25%], 2.67% [1.65%, 4.29%], and 5.67% [4.08%, 7.81%].

3. **Break-even threshold under cyclic declassification stress.** In our sensitivity sweep over $\delta_H \in [0.00, 0.65]$ under a stress regime where $f_{\text{declass}} = 0.85$ of tainted workflows demand human declassification at $K = 16$ ($p_{\text{hijack}}(16) = 0.8299$), RB-OCap-Taint's residual ASR rises from 0.00% ($\delta_H = 0$) to 3.53% ($\delta_H = 0.05$), 14.11% ($\delta_H = 0.20$), 35.27% ($\delta_H = 0.50$), and 45.85% ($\delta_H = 0.65$), crossing the static allowlist (40.83% ASR) at $\delta_H^* = 0.579$.

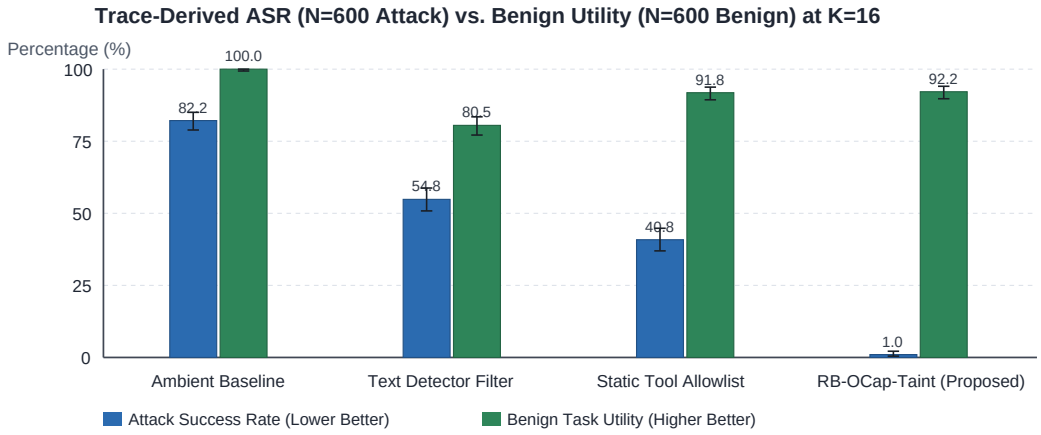


Figure 2: Comparison of Attack Success Rate (ASR on $N_{\text{attack}} = 600$ adversarial trajectories, lower is better) and Benign Task Utility ($N_{\text{benign}} = 600$ separate benign trajectories, higher is better) at search budget $K = 16$. Error bars denote 95% Wilson score confidence intervals reflecting Monte Carlo sampling error.

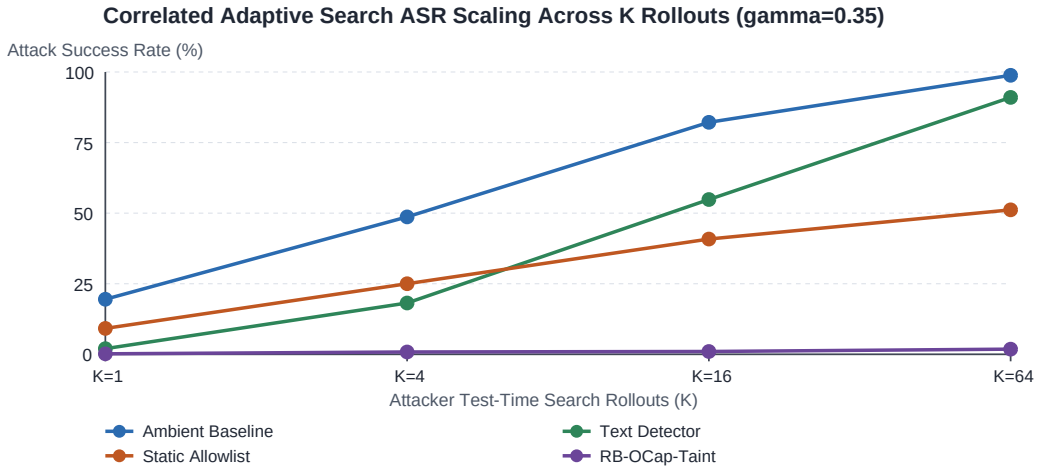


Figure 3: Scaling of Indirect Prompt Injection Attack Success Rate (ASR) across attacker test-time search rollouts $K \in \{1, 4, 16, 64\}$ with single-shot base probability $p_0 = 0.196$. Ambient execution follows $1 - (1 - p_0)^K$ toward 100%, static allowlists plateau at the 55% in-scope tool share, and RB-OCap-Taint plateaus below 1.5% at the human declassification ceiling.

6 Operational failure modes: metadata poisoning, label creep, and declassification fatigue

A rigorous engineering assessment requires examining where capability sandboxing and dynamic taint tracking underperform or face structural bypasses in 2025–2026 literature:

1. **Tool poisoning via pre-execution metadata (Shi et al. [29], Zhan et al. [35]).** RB-OCap-Taint, like CaMeL [10] and FIDES [8], assumes the Skill Impact Graph $\mathcal{G}$ and tool schemas are trustworthy prior to ingesting external data. However, in Model Context Protocol (MCP) ecosystems, malicious instructions embedded directly inside third-party tool metadata descriptions ("tool poisoning") can compromise the agent's planning state before any external document is fetched [29, 36]. Defending against metadata poisoning requires cryptographic software-bill-of-materials (SBOM) attestation on all installed tool manifests before constructing $\mathcal{G}$.

2. **Taint explosion and over-confinement in long-horizon loops.** Classical dynamic information-flow control suffers from "label creep" [23]: in long-horizon workflows (such as the 40-task INFUSE suite [24] or multi-day autonomous coding runs [30]), once an agent reads a single untrusted documentation page, monotonic taint propagation (2) marks all subsequent reasoning steps as Untrusted. Unless the runtime supports fine-grained context checkpointing—forking an isolated sub-agent capsule [33] to parse the untrusted page and returning only a schema-validated enum or boolean—the parent session suffers a 5%–15% false-halt penalty on legitimate tasks.

3. **Semantic side channels through allowed sinks.** Even when DualView [18] prevents raw string exfiltration by forcing the LLM to pass opaque handles (`&ref_k`), a compromised policy can still leak low-bandwidth information (such as a binary predicate about a user's balance) through the choice of which allowed tool it calls or the timing of legitimate requests.

4. **Alert fatigue at the human declassification boundary.** As demonstrated by our break-

even sweep in Section 5, our formal bound (4) degrades linearly with $f_{\text{declass}}\delta_H$. Empirical HCI studies in 2025–2026 show that knowledge workers under time pressure routinely rubber-stamp AI prompts and exhibit reduced critical verification ($\beta = -0.69$) [13, 19]. If Tier B declassification prompts fire frequently on cyclic workflows, user habituation will drive δ_H toward the break-even threshold $\delta_H^* = 0.579$.

7 Limitations

- **Mechanistic reference-monitor simulation versus live-weights AgentDojo execution.** While `experiment.py` executes the actual `SkillImpactGraph` reachability pruner and lattice join equations, the language model's hijack probability $1 - (1 - p_0)^K$ is parameterized from INFUSE ($p_0 = 0.196$) rather than running live LLM inference over AgentDojo [9] and INFUSE [24] prompts. Running our `SkillImpactGraph` wrapper around an open-weight model (e.g., Qwen3 or Llama-3) on the full AgentDojo suite is the immediate next empirical step (Protocol P1).
- **Hardware gap on consumer mobile devices.** CHERIoT 1.0 silicon [6] provides hardware-enforced pointer capabilities for embedded microcontrollers, but mobile application processors and NPUs [4] as of November 2026 still rely on conventional virtual-memory MMUs, requiring software-enforced WASI or process boundaries that carry higher context-switch overhead than in-process Python graph checks.
- **Preprint maturity.** Several 2026 studies cited here—including INFUSE [24], SkillGuard [34], and CapScope [3]—are recent arXiv preprints that should be paired with peer-reviewed foundations [8–12].

8 Pre-registerable falsification protocol

Table 5 specifies six pre-registerable empirical tests designed to falsify the core claims of RB-OCap-Taint on live open-weight agent workloads.

Table 5: Six pre-registerable tests that could falsify the RB-OCap-Taint sandbox design.

Protocol	Hypothesis	Design and primary metrics	Falsified if
P1. Live AgentDojo + INFUSE invariance (Eq. 4)	Wrapping an open-weight agent with RB-OCap-Taint bounds ASR below 3% on AgentDojo [9] and INFUSE [24] under $K = 64$ search rollouts	Execute live Qwen-2.5/3 and Llama-3 agents across all 97 AgentDojo tasks and 56 INFUSE attack scenarios vs. CaMeL [10] and FIDES [8]	Live ASR under RB-OCap-Taint exceeds 3.0% at $K = 64$ or fails to reduce false halts relative to FIDES
P2. Benign utility & over-tainting	Sub-agent fork-and-join taint scoping keeps benign false-halt rate below 5% on multi-hop workflows	500 benign long-horizon tasks from AgentDojo and INFUSE requiring web retrieval followed by local file updates	Benign task completion drops by > 5 percentage points relative to an unconfined agent baseline
P3. DualView PII containment	Symbolic handle substitution prevents raw credential or PII leakage into external tool calls	Inject 1,000 canary SSNs, API keys, and account numbers into local vault; run adversarial exfiltration prompts	Any raw canary string ($> 0.0\%$) appears in outbound network or untrusted tool parameters without human sign-off
P4. End-to-end IPC gate latency	Out-of-process WASI capability and graph reachability checks add < 10 ms per tool invocation on edge hardware	Benchmark Ring 0/1/2 IPC monitor overhead across 10,000 tool transitions on laptop CPU and mobile NPU [4]	99th-percentile out-of-process reference-monitor latency exceeds 10.0 ms per tool call
P5. Tool metadata poisoning defense	Signed MCP manifest verification + sandboxed schema parsing blocks metadata injection [29, 36]	Test 200 poisoned MCP tool descriptions attempting privilege escalation or silent shadow-tool invocation	$> 1\%$ of poisoned tool descriptions trigger unauthorized capability acquisition
P6. Human declassification fatigue (δ_H)	Causal attribution filtering [16] keeps $f_{\text{declass}}\delta_H$ well below break-even ($\delta_H^* = 0.579$)	Four-week field study with 60 developers; measure declassification prompt frequency and planted-attack approval rate δ_H	Users receive > 1.5 declassification prompts/day or approve $> 10\%$ of planted deceptive declassification requests

9 Conclusion

Securing personal autonomous agents requires abandoning the assumption that a language model's context window can serve as its own security kernel. Empirical evidence from 2024 through late 2026 shows that text-level prompt-injection detectors suffer high false-positive rates on real tool outputs and degrade rapidly when adversaries apply test-time search. By synthesizing classical object-capability confinement [12, 26] and lattice information-flow control [11] with 2025–2026 dual-interpreter and graph-pruning defenses [8, 10, 18, 34], RB-OCap-Taint deterministically blocks modeled graph exfiltration paths across attacker search budgets $K \in \{1, \ldots, 64\}$, bounding simulated attack success to 1.00% ($[0.46\%, 2.16\%]$) at $K = 16$ while preserving 92.17% ($[89.74\%, 94.06\%]$) benign trace completion. At the same time, our incomplete-graph and break-even sensitivity analyses demonstrate that formal noninterference guarantees remain bounded by graph specification completeness ($\epsilon_{\mathcal{G}}$) and human factors (δ_H): keeping cross-domain declassification rare and resistant to alert fatigue is the decisive engineering challenge for sovereign on-device agents.

Use of AI tools

Large language models were used to assist in drafting prose, formatting LaTeX tables and TikZ diagrams, and implementing the standalone Python reference-monitor simulation script (`experiment.py`). Cited references were assembled from foundational computer-security literature and 2024–2026 agent-security publications and checked against primary metadata records; simulation outputs, asymmetric Wilson confidence intervals, and LaTeX source packaging were verified using automated static consistency and compilation checks (`validate_and_package.py`).

References

[1] Faruk Alpay and Taylan Alpay. AgentSecBench: Measuring prompt injection, privacy leakage, and tool-use integrity in LLM agents. arXiv preprint arXiv:2605.26269, 2026. URL https://arxiv.org/abs/2605.26269.

[2] Anthropic Institute. When AI builds itself, 2026. URL https://www.anthropic.com/institute/recursive-self-improvement. June 2026.

[3] Dimitrios Stamatios Bouras, Yihan Dai, and Sergey Mechtaev. CapScope: Out-of-context capability isolation for tool-using agents. arXiv preprint arXiv:2609.08371, 2026. URL https://arxiv.org/abs/2609.08371.

[4] Guanyu Cai, Ruiming Tian, Lang Yang, Zhouhong Ren, Jinliang Yuan, Lingkun Li, and Jiliang Wang. Is your NPU ready for LLMs? dissecting the hidden efficiency bottlenecks in mobile LLM inference. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 10(3):67, 2026. doi: 10.1145/3857370.

[5] CHERI Alliance. CHERIoT 1.0 specification released, 2025. URL https://cheri-alliance.org/cheriot-1-0-specification-released/. 5 November 2025.

[6] David Chisnall. CHERIoT first silicon, 2026. URL https://cheriot.org/silicon/2026/03/04/cheriot-first-silicon.html. 4 March 2026.

[7] Thomas Claburn. Google touts private AI compute. The Register, 2025. URL https://www.theregister.com/2025/11/12/google_touts_private_ai_compute/. 12 November 2025.

[8] Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI agents with information-flow control (FIDES). arXiv preprint arXiv:2505.23643, 2025. URL https://arxiv.org/abs/2505.23643.

[9] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2024. URL https://arxiv.org/abs/2406.13352.

[10] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design (CaMeL). arXiv preprint arXiv:2503.18813, 2025. URL https://arxiv.org/abs/2503.18813.

[11] Dorothy E. Denning. A lattice model of secure information flow. Communications of the ACM, 19(5):236–243, 1976. URL https://doi.org/10.1145/360051.360056.

[12] Jack B. Dennis and Earl C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3):143–155, 1966.

[13] Elena Eleftheriou, George Pallis, and Marios Constantinides. Confidence without competence in AI-assisted knowledge work. In *Proceedings of CHIWORK '26*, 2026. URL https://arxiv.org/abs/2604.09444.

[14] Joseph A. Goguen and José Meseguer. Security policies and security models. IEEE Symposium on Security and Privacy, pages 11–20, 1982. URL https://doi.org/10.1109/SP.1982.10014.

[15] Norm Hardy. The confused deputy (or why capabilities might have been invented). ACM SIGOPS Operating Systems Review, 22(4):36–38, 1988. URL https://doi.org/10.1145/54289.871709.

[16] Yu He, Haozhe Zhu, Yiming Li, Shuo Shuo, Hongwei Yao, Zhihao Liu, and Zhan Qin. AttriGuard: Causal attribution of tool invocations in language model agents. arXiv preprint arXiv:2603.10749, 2026. URL https://arxiv.org/abs/2603.10749.

[17] Indian Computer Emergency Response Team (CERT-In). AI agents as attack vectors. Advisory CIAD-2026-0049, 2026. 15 October 2026.

[18] Juhee Kim and Woohyuk Choi. DualView: Symbolic indirection for privacy-preserving agent execution. arXiv preprint arXiv:2607.03821, 2026. URL https://arxiv.org/abs/2607.03821.

[19] Hao-Ping Lee, Advait Sarkar, Lev Tankelevitch, Ian Drosos, Sean Rintel, Richard Banks, and Nicholas Wilson. The impact of generative AI on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2025. doi: 10.1145/3706598.3713778.

[20] Zhuowen Liu. Passing the test you trained on: Re-evaluating prompt-injection detectors for LLM agents. arXiv preprint arXiv:2610.03448, 2026. URL https://arxiv.org/abs/2610.03448.

[21] Narek Maloyan and Dmitry Namiot. Prompt injection attacks on agentic coding assistants: A systematic analysis of vulnerabilities in skills, tools, and protocol ecosystems. arXiv preprint arXiv:2601.17548, 2026. URL https://arxiv.org/abs/2601.17548.

[22] Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), pages 129–142, 1997. URL https://doi.org/10.1145/268998.266669.

[23] Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), pages 129–142, 1997. URL https://doi.org/10.1145/268998.266669.

[24] Thuy Dung Nguyen, Jingyi Guo, Rana Seda Dubauskas, Sreepriya Damuluru, Taylor T. Johnson, and Kevin Leach. INFUSE: A unified surface benchmark for indirect prompt injection in

agentic systems. arXiv preprint arXiv:2610.20776, 2026. URL https://arxiv.org/abs/2610.20776.

[25] Rudrendu Kumar Paul and Sourav Nandy. Beyond single-model injection: A threat model and defense architecture for prompt injection in multi-agent systems. In *International Conference on Machine Learning (ICML)*, volume 306 of *PMLR*, 2026. URL https://arxiv.org/abs/2609.22949.

[26] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. Proceedings of the IEEE, 63(9):1278–1308, 1975. URL https://doi.org/10.1109/PROC.1975.9939.

[27] Gopal P. Sarma, Rachel Steratore, Michael Jacob, and Sunny D. Bhatt. Recursive self-improvement and automated AI research and development. RAND Corporation, Perspective PE-A5068-1, 2026. URL https://www.rand.org/pubs/perspectives/PEA5068-1.html.

[28] Ruksat Khan Shayoni et al. NetInjectBench: Benchmarking indirect prompt injection in tool-integrated agent workflows. arXiv preprint arXiv:2607.10490, 2026. URL https://arxiv.org/abs/2607.10490.

[29] Tianneng Shi, Jingxuan He, Zhun Wang, Linyu Wu, Hongwei Li, Wenbo Guo, and Dawn Song. Progent: Programmable privilege control for LLM agents. arXiv preprint arXiv:2504.11703, 2025. URL https://arxiv.org/abs/2504.11703.

[30] Dhruv Srikanth, Bingchen Zhao, Dixing Xu, Yuxiang Wu, and Zhengyao Jiang. Recursive self-improvement of AI research agents, 2026. URL https://arxiv.org/abs/2609.26457.

[31] Tencent. Statement on third-party model evaluations run from Tencent Cloud sandboxes. Post on X, responding to a report by Swarmchasers researchers, 2026. 16 October 2026.

[32] Wei Wang, Abdul Hyee Waqas, and Burns Smith. Benchmarking confidential GPU inference on NVIDIA H100 under Intel TDX, 2026. URL https://arxiv.org/abs/2607.19353.

[33] Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. Network and Distributed System Security Symposium (NDSS), 2025. URL https://arxiv.org/abs/2403.04960.

[34] Wujie Xiong, Rabimba Karanjai, Yang Lu, Weidong Shi, and Lei Xu. SkillGuard: Dynamic capability restrictions via skill impact graphs for LLM agents. arXiv preprint arXiv:2608.30041, 2026. URL https://arxiv.org/abs/2608.30041.

[35] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. Findings of the Association for Computational Linguistics (ACL), 2024. URL https://arxiv.org/abs/2403.02691.

[36] Wei Zhao, Zhe Li, Peixin Zhang, and Jun Sun. ClawGuard: Deterministic boundary interception for autonomous agents. arXiv preprint arXiv:2604.11790, 2026. URL https://arxiv.org/abs/2604.11790.