State of the Art & Critical Gaps in AI Agent Security

Operating System Realities, Tool Poisoning, and Ambient Authority Across Windows and macOS

Author: Christopher L. T. Brown (clbrown@techpathways.com), Technology Consulting in Digital Forensics, A InfoSec & AI, September 2026

Executive Summary

Autonomous artificial intelligence agents have migrated rapidly from sandboxed browser chat interfaces into direct operating system execution environments. In developer workstations, enterprise administration consoles, and client desktops, agents now interpret high-level human directives into native command execution, file system manipulation, API consumption, and inter-process communication. While this operational shift unlocks major workflow efficiencies, it exposes an unresolved architectural deficit: current operating system security abstractions were engineered for human operators and static binaries, not autonomous probabilistic decision engines.

When an autonomous agent operates on a host, it inherits the ambient authority of the logged-in user. Every security control designed to segregate untrusted code from system resources—such as user session isolation, discretionary access control, and endpoint detection heuristics—assumes that authorized user credentials correlate directly with intentional actions. In an agentic environment, that premise breaks down. An attacker who manipulates the contextual inputs of an agent does not need to compromise the underlying operating system kernel or crack password hashes; the attacker simply directs the legitimate, authenticated agent runtime to execute malicious objectives on their behalf.

1. Ambient Operating System Authority and Boundary Disagreement: Autonomous agent runtimes execute with full user session privileges, granting them unfettered read access to sensitive local data stores. On Microsoft Windows, features like Recall maintain unencrypted or DPAPI-accessible SQLite vector databases containing comprehensive visual and text history [26], [27]. On Apple macOS, local Apple Intelligence databases and Spotlight caches expose personal geolocation, media metadata, and identity tokens to any process possessing the host application's Transparency, Consent, and Control (TCC) entitlements [33]. Crucially, platform vendors frequently reject vulnerability reports regarding local data extraction by arguing that same-context execution does not cross a defined security boundary [27]. This leaves enterprise defenders stranded between vendor boundary definitions and real-world post-exploitation mechanics [13].

2. Unsanitized Tooling Interfaces and Supply-Chain Fragility: The Model Context Protocol (MCP) and integrated development environment (IDE) plugin ecosystems have standardized how language models discover and invoke external tools. However, tool metadata—including schema descriptions, parameter documentation, and operational manuals—is ingested directly into the model's primary prompt context without sanitization. Tool Description Poisoning achieves near 100% attack success rates against frontier reasoning models [9]. Furthermore, remote command execution and DNS-rebinding vulnerabilities in MCP servers allow external websites and untrusted code repositories to compromise developer hosts silently [41], [43], [44].

3. Ephemeral Delegation and the Absence of Cryptographic Provenance: Modern enterprise authentication protocols, including OAuth 2.0 and OpenID Connect, authenticate the human operator to an initial service, but they fail to verify or constrain multi-hop agent delegation. When an agent invokes downstream sub-agents or third-party web services, the execution context loses the original user's intent, creating confused-deputy exposures where poisoned agents execute unauthorized financial transactions, repository modifications, or credential exfiltration [19], [22], [23]. Without tamper-evident authorization chains anchored outside the host runtime, enterprise environments cannot establish non-repudiation or detect autonomous privilege escalation [25], [46].

Figure 1: Core operational failure rates across AI agent toolchains, prompt defenses, and execution policy denylists.

To establish defensive viability across Windows and macOS endpoints, organizations must treat agent processes as untrusted, semi-autonomous actors rather than trusted extensions of the human user. Bridging this gap requires moving beyond brittle prompt filtering toward hardware-enforced monotonic sandboxing, formal protocol separation between tool instructions and data, and off-host cryptographic delegation tokens that bind every tool call to a verified human principal.

1. Industry Tooling Realities and Practical Gaps

1.1 Microsoft Windows: Copilot, Recall Telemetry, and Ambient Credential Extraction

Enterprise Windows environments represent the primary deployment ground for autonomous workplace agents, yet Windows security architecture remains anchored to user-token authorization models conceived decades ago. The introduction of local AI runtimes, specifically Microsoft Copilot and Windows Recall, introduces acute forensic and containment challenges. Recall continuously records screenshots of active desktop sessions, processing visual frames through on-device optical character recognition (OCR) and storing semantic representations within local SQLite vector databases.

Independent security evaluations demonstrated that unprivileged malware operating within the user session can extract the entire historical database of screen captures, passwords, and correspondence without administrative privileges or kernel modifications [26]. Although Microsoft subsequently updated Recall's architecture to require Windows Hello authentication and biometric enrollment for database access [28],

the core operational problem persists: when an authorized user unlocks their session, any agent running within that security context inherits the ability to read and query the local vector store. Microsoft formally designated this extraction technique as falling outside its vulnerability servicing criteria, noting that code executing inside the user's logged-on context is not considered a boundary violation [27].

This boundary denial underscores an operational disconnect between platform vendors and defensive engineers. In practical intrusion scenarios, post-exploitation tooling targets the Data Protection API (DPAPI) and Local Security Authority Subsystem Service (LSASS) secrets to dump Wi-Fi keys, saved browser credentials, and remote desktop certificates [29], [30], [31]. Because autonomous coding agents require write access to developer directories and command-line execution rights, a compromised agent becomes an automated, resident credential-harvesting engine that evades endpoint detection and response (EDR) agents because its execution flow originates from trusted developer binaries [32].

1.2 Apple macOS: Apple Intelligence Caches, TCC Inheritance, and Scripting Evasion

On macOS, Apple has constructed a privacy model centered around Transparency, Consent, and Control (TCC) and sandboxed application bundles. However, the integration of on-device Apple Intelligence introduces data stores that bypass conventional user-facing permission prompts when accessed via trusted system utilities.

A notable demonstration of this vulnerability is the 'Sploitlight' exploit disclosed by Microsoft Threat Intelligence, which demonstrated that malicious actors could abuse Spotlight metadata importer plugins to bypass TCC restrictions and siphon Apple Intelligence caches without triggering consent dialogs [33]. These caches aggregate sensitive user artifacts, including high-precision geolocation trails, photo facial recognition tags, and indexed conversational search history. This vulnerability followed a long succession of macOS TCC inheritance flaws, such as powerdir and HM-Surf [34], as well as recent vulnerabilities across developer applications like Cursor and terminal tools [35].

Beyond data cache leakage, macOS presents a unique execution-monitoring gap through AppleScript and the Open Scripting Architecture (OSA). While basic osascript execution may trigger security alerts in enterprise monitoring, AppleScript-Objective-C bridges scripting commands directly into native Cocoa runtime frameworks. Researchers have demonstrated that an agent script can silently monitor the system clipboard via NSPasteboard and launch hidden processes via NSWorkspace without generating events in Apple's Endpoint Security Framework (ESF) [36], [37]. While Apple's Private Cloud Compute (PCC) establishes a rigorous, cryptographically verifiable remote inference environment with non-targetability guarantees [38], [39], [40], PCC protects data exclusively in transit and during remote compute; it does not protect the client endpoint from on-device agent compromise.

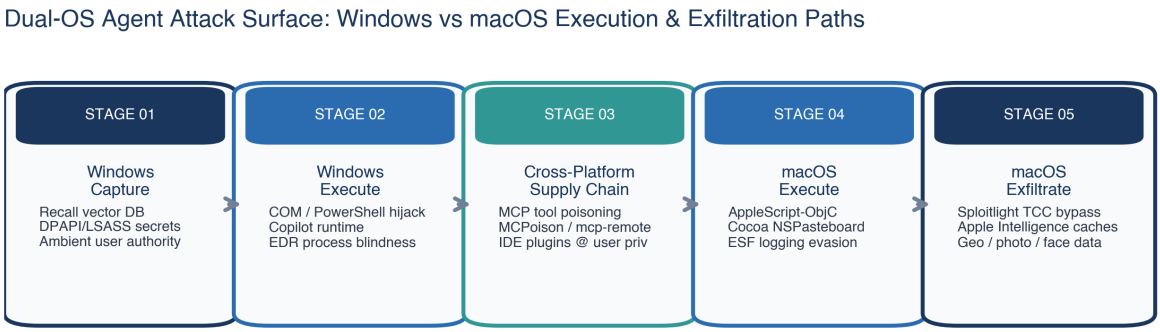

Figure 2: Dual-OS attack surface workflow illustrating Windows and macOS execution and data exfiltration vectors.

1.3 Developer Toolchains, MCP Infrastructure, and Judicial Admissibility

The rapid adoption of the Model Context Protocol (MCP) across developer environments has created a decentralized, unvetted attack surface. Research published by the Cloud Security Alliance indicates that MCP tool poisoning achieves a 36.5% average attack success rate across twenty commercial language models, rising to 72.8% on specialized reasoning models [41]. Specific vulnerabilities illustrate the severity of this exposure: CVE-2025-54136 ('MCPoison') enabled silent MCP server redirection within the Cursor IDE [42], CVE-2025-6514 in mcp-remote allowed remote command injection affecting an estimated 500,000 developer workstations [43], and CVE-2025-66414 demonstrated that DNS rebinding can convert local loopback MCP servers into accessible endpoints for malicious web pages [44].

Simultaneously, malicious IDE extensions targeting JetBrains and Visual Studio platforms have harvested GitHub tokens, AWS credentials, and 1Password secrets under the guise of AI productivity plugins [45]. The Open Worldwide Application Security Project (OWASP) recognized these emergent structural risks by publishing the OWASP Top 10 for Agentic Applications 2026, ranking Agent Goal Hijacking (ASI01), Tool Misuse (ASI02), and Identity and Privilege Abuse (ASI03) as the most critical threats [46], expanding upon earlier warnings regarding Excessive Agency [47].

From a digital forensics and legal admissibility standpoint, autonomous agent activity introduces severe chain-of-custody complications. In landmark evidentiary rulings such as Lorraine v. Markel (241 F.R.D. 534, D. Md. 2007) and R v. Hamdan (2017 BCSC 676), judicial courts established that electronically stored information (ESI) must satisfy rigorous tests of authenticity, system integrity, and verifiable human agency. When an autonomous agent modifies host configurations, compiles binaries, or accesses cloud storage, it generates thousands of ephemeral file system and registry events. If those actions cannot be deterministically tied to a specific human command and verified via tamper-evident logs, the resulting evidence risks judicial exclusion, leaving organizations unable to prove the provenance or integrity of their internal data.

2. Academic Literature Review and Solid Science Foundations

2.1 The Foundational Fragility of Prompt and Planning Defenses

Academic literature has demonstrated that indirect prompt injection is an unsolved architectural flaw inherent in systems where data and program control flow are processed across the same linguistic channel. Greshake et al. established this vulnerability by showing that untrusted third-party data can hijack language model execution paths across search engines and automated code completion utilities [1]. Subsequent evaluations by Zhan et al. tested eight prominent prompt-filtering defenses and found that every single defense bypassed under adaptive adversarial evaluation, with attack success rates exceeding 50% [2]. This finding proves that static defense benchmarks provide a false sense of security when confronted with adaptive adversaries.

Rather than mitigating this fragility, advanced model reasoning often exacerbates it. Ferrag et al. cataloged over thirty attack techniques spanning host-to-tool and agent-to-agent communication, identifying the 'Toxic Agent Flow' exploit where prompt injections cascade across autonomous workflows [3]. In empirical testing of multi-agent planning frameworks, Wang discovered through the PlanFlip benchmark that frontier models like GPT-5 exhibited higher planning-phase injection success rates (0.68) than smaller models, directly refuting the assumption that superior reasoning capabilities inherently confer security robustness [4].

Furthermore, an agent's memory architecture serves as a stealthy vector for persistent compromise. Dong et al. introduced the MINJA attack framework, demonstrating that adversaries can poison an agent's long-term memory through passive query interactions without direct system access [5]. Complementary research into skill-based execution, including SkillJect [6], Poise [7], and EVOMAL [8], reveals that malicious instructions injected into stored skill files can lie dormant until triggered by unrelated user prompts, surviving system restarts and skill re-indexing.

2.2 Tool Description Poisoning, MCP Benchmarks, and Execution Balkanization

When an agent interacts with external systems via MCP, it selects appropriate functions by reading natural language descriptions provided by the server. Liu et al. systematically benchmarked this interaction and demonstrated that Tool Description Poisoning (TDP)—embedding adversarial jailbreaks and extraction commands within tool description metadata—achieves near 100% attack success rates against state-of-the-art models like GPT-4o [9]. Crucially, the authors demonstrated that applying prompt-level guardrails to tool descriptions is counterproductive, a phenomenon termed the 'Firewall Fallacy', because guardrail filtering disrupts legitimate schema parsing without stopping obfuscated payloads.

In a comprehensive systematization of 39 execution-security papers published between 2023 and 2026, Rashidi identified severe balkanization across the research community [10]. Studies examining policy enforcement report that operational denylists suffer a 69% to 98% real-world failure rate when tested against adversarial obfuscation. Yet, isolation and sandboxing papers routinely assume robust policy enforcement without re-evaluating their proposed architectures under hostile conditions. Hossain et al.'s PRISMA systematic review of 85 agent security studies confirmed this imbalance, showing that published offensive research outpaces defensive literature by 3.9 to 1 [11]. Most critically, action-layer

vulnerabilities—such as tool misuse, parameter tampering, and sandbox evasion—are addressed in only 4.7% of papers despite presenting the highest real-world catastrophe potential.

2.3 Privilege Escalation, Command-Line Harnesses, and Autonomous Propagation

At the system execution layer, language models function as capable privilege escalation engines. Happe and Cito evaluated autonomous agent performance against Linux privilege escalation challenges in their hackingBuddyGPT framework, finding that GPT-4-Turbo successfully identified and exploited 33% to 83% of privilege misconfigurations, matching the capability of experienced human penetration testers [12]. When deployed as command-line assistants, agents exhibit high susceptibility to jailbreaks embedded in project documentation; Luo et al.'s AdvCLI benchmark demonstrated that CLI agents frequently proceed beyond safety refusal when processing malicious instructions hidden within README files and shell scripts [16]. Similarly, Lee et al. achieved a 41.3% attack success rate against multimodal computer-use agents by bypassing visual safety alignment using terminal terminal injection techniques [17].

The risk of autonomous agent compromise escalates dramatically when self-propagation primitives are introduced. Zhang et al. developed AgentWorm, demonstrating that a single poisoned context can trigger a self-propagating infection across connected agent networks, achieving a 63% aggregate attack success rate across five distinct language model architectures [18]. The researchers confirmed cross-framework transferability into diverse agent architectures, including Hermes Agent, proving that autonomous propagation is an inherent weakness of current agent execution loops rather than a flaw in an isolated runtime.

In response to these execution threats, recent defensive literature emphasizes formal least-privilege containment. Shi et al. proposed Progent, an architecture that enforces monotonic privilege narrowing using Satisfiability Modulo Theories (SMT) solver-gated policy updates, cutting attack success rates while preserving operational utility [13]. Similarly, Hu et al.'s AgentSentinel [14] and Gong et al.'s CSAgent [15] introduce real-time reference monitors that intercept operating system API calls, isolating computer-use agents within restricted context spaces.

2.4 Cryptographic Delegation, Off-Host Authorization, and Identity Chains

To prevent confused-deputy attacks across distributed multi-agent systems, researchers are transitioning away from centralized OAuth tokens toward decentralized cryptographic identity. Huang et al. [19], Xu et al. [20], and Rodriguez Garzon et al. [21] established that Decentralized Identifiers (DIDs) and W3C Verifiable Credentials provide the requisite mathematical foundation for ephemeral, verifiable agent identities that do not rely on static API secrets.

Building upon cryptographic identity, recent architectures shift authorization logic entirely off the agent host. Kodathala developed aiAuthZ, which binds agent identity to HMAC signatures and validates actions through an off-host policy evaluation engine with hash-chained auditing, reducing residual attack success rates to 0% across fifteen evaluated models [22]. Muruaga formulated the Agentic Principal Chain (APC), demonstrating that bounding an agent's authority through an immutable cryptographic chain eliminates data exfiltration risks in benchmarks like AgentDojo [23]. These architectures are complemented by Gallo's Proof-of-Continuity model for temporal authority propagation [24] and Stefanescu et al.'s Lineage protocol

extension for User-Managed Access (UMA 2.0), which enforces fine-grained policy propagation across complex, multi-hop agent delegation sequences [25].

3. Consolidated Investigative Gap Matrix

A systematic synthesis of empirical vendor telemetry, judicial evidence standards, and academic literature surfaces ten critical capability gaps across contemporary AI agent architectures. These gaps represent the primary failure points that permit prompt injection, privilege escalation, and unauthorized data extraction across Windows and macOS environments.

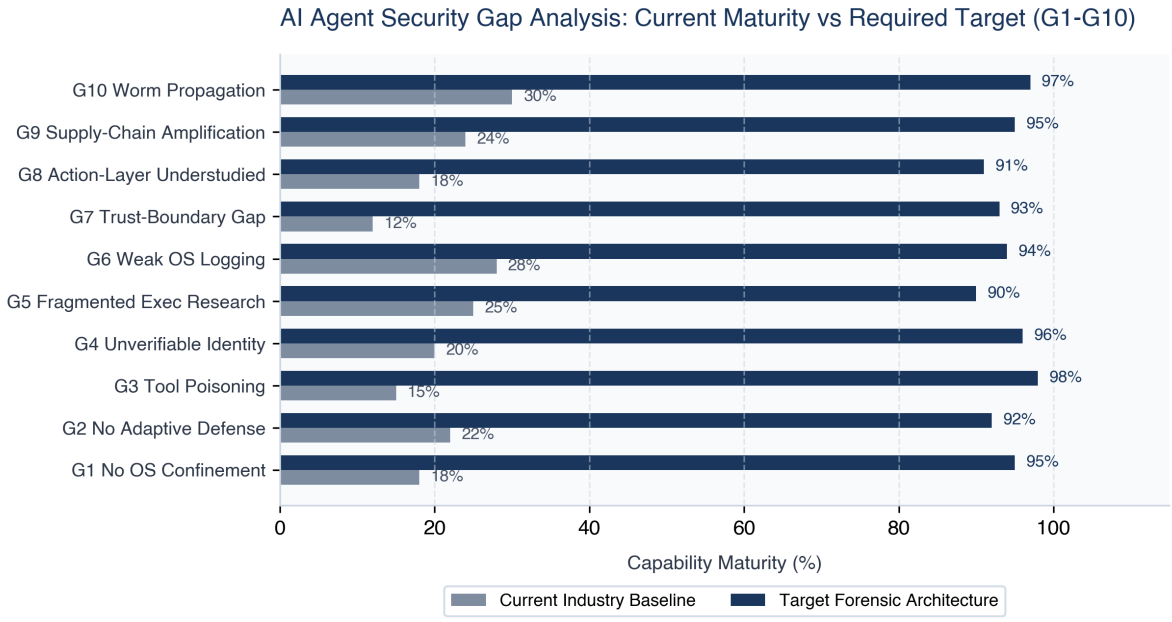

Figure 3: Capability maturity gap analysis across ten operational domains (G1–G10), contrasting current defenses with target requirements.

ID	Functional Area	Specific Capability Gap	Concrete Investigative & Operational Impact	Core References
G1	OS-Native Confinement	Absence of native OS confinement for agent runtimes; processes inherit ambient user credentials and permissions.	Malware or poisoned agents in the user context drain Windows Recall vector stores and macOS Apple Intelligence caches without crossing vendor-recognized security boundaries.	[26], [27], [33], [13]
G2	Adaptive Prompt Defense	Lack of adversarial-aware, adaptive prompt defense mechanisms; existing filters rely on static benchmarks.	Over 50% adaptive attack success rate; perimeter prompt filters provide false operational assurance against determined	[2], [16], [9]

			adversaries.	
G3	Tool Schema Sanitization	Tool description and metadata channels remain unverified and unsanitized within MCP and agent frameworks.	Tool Description Poisoning achieves 36.5% to 100% attack success rates; silent SSH key extraction occurs through compromised utility tool definitions.	[9], [41], [6], [7]
G4	End-to-End Identity	Inability to verify end-to-end cryptographic identity and human intent across multi-hop agent delegation.	Confused-deputy vulnerabilities persist; hijacked authorized agents pass standard authentication checks and execute unauthorized actions.	[19]–[25], [46]
G5	Execution Research	Execution-security research remains fragmented across isolated silos without standardized adversarial benchmarks.	Policy denylists suffer 69% to 98% real-world failure rates; enterprise defenders cannot validate vendor containment claims.	[10], [11]
G6	Host Audit Telemetry	Host telemetry and endpoint sensors are blind to fileless, in-memory scripting and agent-mediated API calls.	Compromised agent actions using AppleScript-ObjC or PowerShell evade Endpoint Security Framework (ESF) and EDR detection; audit logs fail court admissibility tests.	[36], [37], [32]
G7	Trust Boundary Alignment	Fundamental disagreement between platform vendors and enterprise defenders regarding trust boundaries.	Vendors reject local agent data extraction as non-vulnerabilities; accountability stalls and security patches are deprioritized.	[26], [27], [28]
G8	Action-Layer Security	Disproportionate research focus on model generation while neglecting action-layer containment and sandbox escapes.	Action-layer vulnerabilities appear in only 4.7% of academic literature despite representing the highest real-world destruction potential.	[11], [12], [17]
G9	Supply Chain Isolation	IDE extensions, local MCP servers, and skill packages execute at full user privilege without process sandboxing.	Single compromised repository configs or malicious plugins harvest cloud API keys, SSH credentials, and password vaults across developer fleets.	[41]–[45]

G10	Worm Infection Controls	Absence of operational propagation boundaries and infection-loop controls across interconnected agent swarms.	Self-propagating agent worms achieve 63% infection success across heterogeneous model architectures, turning multi-agent environments into botnets.	[18]

4. Key Takeaways and Architectural Implications

Mitigating the vulnerabilities outlined in this analysis requires fundamentally altering how operating systems, agent runtimes, and developer frameworks treat autonomous execution. Relying on model alignment or prompt filtering to prevent system compromise is ineffective. Organizations must implement defense-in-depth architectures grounded in the following technical imperatives:

1. Monotonic Privilege Narrowing and Hardware Sandboxing: Agent execution runtimes must never execute with ambient user authority. Operating systems must establish dedicated, unprivileged agent execution profiles isolated via hardware virtualization, seccomp filters, or macOS App Sandbox containers. Privilege sets must follow monotonic narrowing models such as Progent [13], where capabilities are permanently shed as task execution progresses, preventing late-stage privilege escalation.

2. Strict Semantic Separation of Tool Metadata and Untrusted Data: The Model Context Protocol and related tooling frameworks must abandon raw text ingestion of tool definitions. Tool descriptions must be parsed as structured, validated schemas with cryptographic signatures verifying developer provenance. Dynamic parameters must be strictly segregated from model system instructions using isolated execution enclaves, eliminating Tool Description Poisoning [9], [41].

3. Off-Host Cryptographic Delegation Chains: Inter-agent workflows must replace static API tokens with decentralized identifiers (DIDs) and ephemeral verifiable credentials. Every downstream action must carry a verifiable authorization chain anchored to a verified human authorization event, as demonstrated in the aiAuthZ [22] and Agentic Principal Chain [23] architectures. If an agent's instruction context diverges from the verified principal's intent, downstream tools must reject the invocation automatically.

4. Kernel-Level Forensic Telemetry and Non-Repudiation: Host telemetry sensors must evolve beyond legacy process monitoring to capture agent-specific execution semantics. Endpoint sensors must record intent-to-action bindings, logging the specific prompt context, tool selection, and executed system call in a tamper-evident, append-only ledger. This telemetry is essential to satisfy judicial admissibility standards under Lorraine v. Markel and R v. Hamdan, ensuring enterprise audits withstand legal scrutiny [36], [32].

5. Mandatory Adaptive Red-Teaming and Infection Loop Breaks: Enterprise security teams must validate agent deployments against adaptive adversarial benchmarks rather than static safety evaluations. Architectures must incorporate circuit-breaker controls that monitor agent communication

graphs for cyclic message passing and anomalous replication, arresting the propagation of self-replicating agent worms before network-wide compromise occurs [2], [18].

> **KEY TAKEAWAY: DEFENSIVE REALIGNMENT FOR AGENTIC WORKLOADS**
> Treating autonomous AI agents as trusted local software is the core systemic vulnerability of modern enterprise endpoints. Security must be enforced outside the language model's cognitive loop: at the operating system boundary via hardware sandboxes, at the network layer via cryptographic delegation proofs, and at the audit layer via tamper-evident intent logging. Only when agent authority is strictly decoupled from user identity can enterprise organizations safely realize the productivity benefits of autonomous AI systems.

References

[1] 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," *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISEC '23),* https://consensus.app/papers/details/667ace18801a59ce89d56d0e636d376c, 2023, Last accessed September 18th 2026

[2] Qiusi Zhan et al., "Adaptive Attacks Break Defenses Against Indirect Prompt Injection Attacks on LLM Agents," *arXiv:2503.00061,* https://consensus.app/papers/details/cadf2fb6ff3b5d138c4b896604a4e7ba, 2025, Last accessed September 18th 2026

[3] Mohamed Amine Ferrag et al., "From Prompt Injections to Protocol Exploits: Threats in LLM-Powered AI Agents Workflows," *ICT Express, DOI: 10.1016/j.icte.2025.12.001,* https://consensus.app/papers/details/4ab586bc9f9358f99651d024e232fe26, 2025, Last accessed September 18th 2026

[4] Yuhang Wang, "PlanFlip: Attacking Multi-Agent LLM Systems via Planning-Phase Prompt Injection," *arXiv:2607.16199,* https://consensus.app/papers/details/82c6f234b2c05e599d0079b140e418bf, 2026, Last accessed September 18th 2026

[5] Shen Dong et al., "Memory Injection Attacks on LLM Agents via Query-Only Interaction," *Advances in Neural Information Processing Systems 38 (NeurIPS 2025),* https://consensus.app/papers/details/e39728a186f35ce0be89fac1f8b1ab76, 2025, Last accessed September 18th 2026

[6] Xiao-Jun Jia et al., "SkillJect: Effectively Automating Skill-Based Prompt Injection for Skill-Enabled Agents," *arXiv:2602.14211,* https://consensus.app/papers/details/c05832b027df58d2b9f223e73565eaf5, 2026, Last accessed September 18th 2026

[7] Haochang Hao et al., "Poise: Position-Aware One-Instruction Skill Injection for Silent Execution on LLM Agents," *arXiv:2606.07943,* https://consensus.app/papers/details/a115140b9c9559a9a3c393e5885ce6c0, 2026, Last accessed September 18th 2026

[8] Xiaodong Wu et al., "EVOMAL: Self-Poisoning in Self-Evolving Coding Agents," *Research Report,* https://consensus.app/papers/details/6dc3672e71ae52e0ac04674be2499e6c, 2026, Last accessed September 18th 2026

[9] Shi Liu et al., "When the Manual Lies: A Realistic Benchmark to Evaluate MCP Poisoning Attacks for LLM Agents," *Proceedings of the 2026 29th International Conference on Computer Supported Cooperative Work in Design (CSCWD), DOI: 10.1109/cscwd68734.2026.11582512,* https://consensus.app/papers/details/2768f983fa445b43a03330f23d08158e, 2026, Last accessed September 18th 2026

[10] Mohammadreza Rashidi, "The Balkanization of Execution-Security Research for AI Coding Agents: Isolation, Access Control, and Time-of-Check-to-Time-of-Use Vulnerabilities," *arXiv:2607.05743,* https://consensus.app/papers/details/0383e07c2fca57b8acca407b9c613cdc, 2026, Last accessed September 18th 2026

[11] Md Jafrin Hossain et al., "On Understanding, Identifying, and Mitigating Vulnerabilities in Agentic Large Language Models," *arXiv:2608.10530,* https://consensus.app/papers/details/52dda524a21054c7b265a860d785cab2, 2026, Last accessed September 18th 2026

[12] Andreas Happe and Jürgen Cito, "LLMs as Hackers: Autonomous Linux Privilege Escalation Attacks," *Empirical Software Engineering, vol. 30, DOI: 10.1007/s10664-025-10758-3,* https://consensus.app/papers/details/13365b526d5a563eb8420c4cbd1de9e1, 2023, Last accessed September 18th 2026

[13] Tianneng Shi et al., "Progent: Securing AI Agents with Privilege Control," *arXiv:2504.11703,* https://consensus.app/papers/details/629a8810148555b785f9e4b1d48f86a8, 2025, Last accessed September 18th 2026

[14] Haitao Hu et al., "AgentSentinel: An End-to-End and Real-Time Security Defense Framework for Computer-Use Agents," *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), DOI: 10.1145/3719027.3765064,* https://consensus.app/papers/details/b026777f622654aa95864718c3961058, 2025, Last accessed September 18th 2026

[15] Haochen Gong et al., "Secure and Efficient Access Control for Computer-Use Agents via Context Space," *arXiv:2509.22256,* https://consensus.app/papers/details/dc37c24d158c5e81b9db088484bcdb53, 2025, Last accessed September 18th 2026

[16] Weidi Luo et al., "Your Harness is Not Secure: Benchmarking Real-world Threat of Command Line Interface Agent," *arXiv:2510.06607,* https://consensus.app/papers/details/7b7bec3343e253b7940ee63d3895d813, 2025, Last accessed September 18th 2026

[17] Sejin Lee et al., "Sudo Rm -rf Agentic_security," *arXiv:2503.20279,* https://consensus.app/papers/details/3a07afde65aa54839f45cd581f4a7209, 2025, Last accessed September 18th 2026

[18] Yihao Zhang et al., "AgentWorm: Self-Propagating Attacks Across LLM Agent Ecosystems," *arXiv:2603.15727,* https://consensus.app/papers/details/186a4e89c5ef579aa7512b71e7f21ceb, 2026, Last accessed September 18th 2026

[19] Ken Huang et al., "A Novel Zero-Trust Identity Framework for Agentic AI: Decentralized Authentication and Fine-Grained Access Control," *Proceedings of the 2026 International Conference on AIxDKE, DOI: 10.1109/aixdke67294.2026.00024,* https://consensus.app/papers/details/ae140942ae0c5c1ab5471f5dfca70aac, 2025, Last accessed September 18th 2026

[20] Ming-Hui Xu et al., "AgentDID: Trustless Identity Authentication for AI Agents," *Proceedings of the 2026 IEEE 46th International Conference on Distributed Computing Systems (ICDCS), DOI: 10.1109/2575-8411.2026.00116,* https://consensus.app/papers/details/22bdc6409f37587db89378210d524419, 2026, Last accessed September 18th 2026

[21] Sandro Rodriguez Garzon et al., "AI Agents with Decentralized Identifiers and Verifiable Credentials," *Science and Technology Publications, DOI: 10.5220/0014234400004052,* https://consensus.app/papers/details/ed0db80a02cf5174be52ae89ca786253, 2025, Last accessed September 18th 2026

[22] Sai Varun Kodathala, "aiAuthZ: Off-Host, Identity-Bound Authorization for AI Agents," *arXiv:2607.05518,* https://consensus.app/papers/details/a3a9fcdc665353ffa208652b1b3be6ac, 2026, Last accessed September 18th 2026

[23] Xabier Muruaga, "Bounded Agents: Delegation Security for Multi-Agent AI Systems," *arXiv:2608.15888,* https://consensus.app/papers/details/3f87674949345bb8b7fae9ceaa488f41, 2026, Last accessed September 18th 2026

[24] Nicola Gallo, "Proof-of-Continuity: A Temporal Model for Authority Propagation in Distributed Systems and AI Agents," *arXiv:2607.08906,* https://consensus.app/papers/details/f3e46a03e8e95983a9c138acea18288e, 2026, Last accessed September 18th 2026

[25] Stefania Stefanescu et al., "Lineage: A UMA2.0 Protocol Extension for Multi-Hop LLM Agent Delegation," *IEEE Access, DOI: 10.1109/access.2026.3708618,* https://consensus.app/papers/details/39445b4a71205b01bbaeb787b78b11f1, 2026, Last accessed September 18th 2026

[26] Michael Hill, "Microsoft's Windows Recall still allows silent data extraction," *CSO Online,* https://www.csoonline.com/article/4159643/microsofts-windows-recall-still-allows-silent-data-extraction.html, 2025, Last accessed September 18th 2026

[27] iTnews, "Microsoft says new Windows Recall bypass isn't a vulnerability," *iTnews Australia,* https://www.itnews.com.au/news/microsoft-says-new-windows-recall-bypass-isnt-a-vulnerability-624918, 2025, Last accessed September 18th 2026

[28] Microsoft Windows Experience Team, "Update on Recall security and privacy architecture," *Windows Blog,* https://blogs.windows.com/windowsexperience/2024/09/27/update-on-recall-security-and-privacy-architecture/, September 27, 2024, Last accessed September 18th 2026

[29] Synacktiv, "Windows secrets extraction: a summary," *Synacktiv Security Research,* https://synacktiv.com/publications/windows-secrets-extraction-a-summary, 2024, Last accessed September 18th 2026

[30] The Hacker Recipes, "DPAPI protected secrets dumping," *Active Directory Security Guide,* https://www.thehacker.recipes/ad/movement/credentials/dumping/dpapi-protected-secrets, 2025, Last accessed September 18th 2026

[31] HackTricks, "Extracting DPAPI Passwords and Credential Vaults," *HackTricks Hardening & Post-Exploitation Guide,* https://github.com/b4rdia/HackTricks/blob/master/windows-hardening/windows-local-privilege-escalation/dpapi-extracting-passwords.md, 2025, Last accessed September 18th 2026

[32] Aembit, "Why Hybrid Windows Environments are Still a Security Blind Spot," *Aembit Identity & Access Engineering,* https://aembit.io/blog/why-hybrid-windows-security-blind-spot/, 2025, Last accessed September 18th 2026

[33] Christine Fossaceca (Microsoft Threat Intelligence), "Sploitlight: Analyzing a Spotlight-based macOS TCC vulnerability," *Microsoft Security Blog,* https://www.microsoft.com/en-us/security/blog/2025/07/28/sploitlight-analyzing-a-spotlight-based-macos-tcc-vulnerability/, July 28, 2025, Last accessed September 18th 2026

[34] Microsoft Threat Intelligence, "New macOS vulnerability 'HM Surf' could lead to unauthorized data access (CVE-2024-44133)," *Microsoft Security Blog,* https://www.microsoft.com/security/blog/2024/10/17/new-macos-vulnerability-hm-surf-could-lead-to-unauthorized-data-access/, October 17, 2024, Last accessed September 18th 2026

[35] CERT Polska, "TCC Bypass vulnerabilities in six applications for MacOS," *CERT.pl Security Advisories,* https://cert.pl/en/posts/2025/08/tcc-bypass/, August 2025, Last accessed September 18th 2026

[36] Phil Stokes, "How AppleScript Is Used For Attacking macOS," *SentinelOne Research,* https://www.sentinelone.com/blog/how-offensive-actors-use-applescript-for-attacking-macos/, 2024, Last accessed September 18th 2026

[37] Red Canary, "AppleScript Threat Detection Report," *Red Canary Threat Intelligence,* https://redcanary.com/threat-detection-report/techniques/applescript/, 2025, Last accessed September 18th 2026

[38] Apple Security Engineering and Architecture (SEAR), "Private Cloud Compute: A new frontier for AI privacy in the cloud," *Apple Security Blog,* https://security.apple.com/blog/private-cloud-compute/, June 10, 2024, Last accessed September 18th 2026

[39] Apple Security Research, "Security research on Private Cloud Compute," *Apple Security Blog,* https://security.apple.com/blog/pcc-security-research/, October 24, 2024, Last accessed September 18th 2026

[40] Apple Inc., "Private Cloud Compute Security Guide," *Apple Documentation,* https://security.apple.com/documentation/private-cloud-compute, 2024, Last accessed September 18th 2026

[41] Cloud Security Alliance, "MCP Attack Surface: Tool Poisoning and IDE Auto-Execution," *CSA Research Note,* https://labs.cloudsecurityalliance.org/research/csa-research-note-mcp-tool-poisoning-auto-execution-20260701/, July 2026, Last accessed September 18th 2026

[42] Check Point Research, "Cursor IDE's MCP Vulnerability (MCPoison, CVE-2025-54136)," *Check Point Research Blog,* https://research.checkpoint.com/2025/cursor-vulnerability-mcpoison/, August 5, 2025, Last accessed September 18th 2026

[43] vulnerablemcp.info, "mcp-remote OS Command Injection (CVE-2025-6514)," *VulnerableMCP Security Advisories,* https://vulnerablemcp.info/vuln/cve-2025-6514-mcp-remote-rce.html, 2025, Last accessed September 18th 2026

[44] GitHub Advisory Database, “Localhost MCP Server DNS Rebinding Vulnerability (CVE-2025-66414),” *GitHub Advisory,* https://github.com/advisories/GHSA-w48q-cv73-mx4w, 2026, Last accessed September 18th 2026

[45] Cloud Security Alliance, “AI Toolchain Hijacked: IDE Plugin API Key Theft,” *CSA Research Note,* https://labs.cloudsecurityalliance.org/research/csa-research-note-ai-developer-toolchain-supply-chain-202606/, June 2026, Last accessed September 18th 2026

[46] OWASP GenAI Security Project, “OWASP Top 10 for Agentic Applications 2026,” *OWASP Foundation,* https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/, 2026, Last accessed September 18th 2026

[47] OWASP GenAI Security Project, “OWASP Top 10 for Large Language Model Applications (2025),” *OWASP Foundation,* https://genai.owasp.org/llm-top-10/, 2025, Last accessed September 18th 2026