CSAI Foundation

Cloud Security Alliance AI Safety Initiative

The AI Agent Disclosure Vacuum

Silent Bug Bounties, Missing CVEs, and the Accountability Gap in Agentic Security

Unofficial AI-assisted Research

2026-04-17

Table of Contents

Executive Summary

The vulnerability disclosure compact that has governed the software security industry for three decades rests on a set of tacit agreements: vendors accept reports, assign identifiers, patch flaws, and disclose outcomes in a roughly standardized sequence. That compact was designed for software with clear ownership, fixed versions, and predictable behavior. Agentic AI systems violate every one of those assumptions.

As of early 2026, security researchers have documented a pattern of vulnerability reports to AI platform vendors that result in no CVE assignment, no public advisory, and no coordinated disclosure — what this paper terms the "silent bounty" problem. Between January and February 2026 alone, researchers filed more than thirty CVEs targeting Model Context Protocol (MCP) servers, clients, and infrastructure, yet independent analysis suggests that figure reflects only a fraction of the exploitable surface [1]. Flowise, one of the most widely deployed open-source AI agent builders, carried a maximum-severity CVSS 10.0 remote code execution vulnerability (CVE-2025-59528) for nearly seven months between its patch and confirmed in-the-wild exploitation, during which an estimated twelve to fifteen thousand instances remained publicly reachable [2, 18]. The EchoLeak vulnerability (CVE-2025-32711) in Microsoft 365 Copilot demonstrated that a zero-click attack requiring no user interaction could silently exfiltrate sensitive organizational data through a simple email [3].

These incidents point to a structural problem rather than a collection of isolated oversights. The Common Vulnerabilities and Exposures (CVE) program was not designed to accommodate emergent, non-deterministic systems composed of model providers, orchestration frameworks, tool registries, and deployer-defined configurations. The result is a disclosure vacuum where accountability diffuses across vendor layers, exploitation windows compress dramatically, and enterprises deploying agents inherit risks they have no systematic means of tracking.

This paper analyzes the anatomy of that vacuum, surveys the case evidence, and advances a set of recommendations for AI platform vendors, enterprise deployers, security researchers, and standards bodies. The urgency is not theoretical: the mean time from vulnerability disclosure to confirmed exploitation across the software industry has fallen to less than one day in some categories [14], and AI-generated exploits can now be produced for known flaws in fifteen minutes at negligible cost [4]. Closing the disclosure gap is a prerequisite for agentic AI operating safely at enterprise scale.

Introduction: The Disclosure Compact Meets Its Limits

Coordinated vulnerability disclosure emerged as a discipline in the 1990s from a pragmatic recognition that neither full immediate disclosure nor indefinite vendor secrecy served users well. The framework that crystallized — report privately, give vendors a bounded remediation window, disclose publicly once a patch exists — reduced exploitation rates for conventional software and established clear norms around CVE identifiers, CVSS scoring, and NVD enrichment. MITRE's CVE program now catalogs tens of thousands of vulnerabilities annually, and organizations depend on NVD feeds, vendor advisories, and bug bounty aggregators to maintain situational awareness across their attack surface.

Agentic AI systems did not arrive gradually enough for disclosure infrastructure to adapt. In 2025, platforms such as Flowise, LangFlow, and dozens of MCP-compatible orchestrators reached production deployment at scale, often in enterprises that had no dedicated security review process for AI components. The Model Context Protocol, introduced in late 2024, created a new class of software component — the MCP server — that connects AI models to external tools, databases, and APIs, and rapidly proliferated across thousands of community-developed implementations. Security analysis of those implementations found that eighty-two percent of surveyed MCP servers use file operations vulnerable to path traversal [1]. The researchers who documented this pattern also noted that the architectural novelty of MCP — where an AI model mediates requests between a client and a server it does not fully control — creates exploitation scenarios that have no direct analog in conventional software, and therefore no established disclosure pathway [1, 20].

The gap between the pace of deployment and the pace of disclosure process development is not unique to AI, but the characteristics of agentic systems amplify its consequences in ways that deserve specific analysis. Agentic systems are compositional: a single production deployment may involve a frontier model, a third-party orchestration framework, a set of community MCP servers, and a deployer-provided configuration that none of the upstream vendors have reviewed. They are non-deterministic: the same input may produce different outputs depending on context, conversation history, and in-context retrieved data, making reproducible vulnerability demonstration difficult. They are persistent: agents often hold credentials, session tokens, and memory that persist across invocations, creating lateral movement surfaces that do not exist in stateless software. And they operate with delegated authority: an agent asked to file an issue, send an email, or execute code acts on behalf of the user with the user's credentials, meaning a compromised agent is not merely a compromised software component but a compromised principal.

Each of these characteristics creates distinct challenges for disclosure processes that assume a clear vendor, a reproducible flaw, and a bounded impact. Understanding those challenges is the precondition for addressing them.

The Structural Mismatch: CVE Was Not Built for AI Agents

The CVE program assigns identifiers to discrete, reproducible weaknesses in specific software components at specific versions. That model works well for a buffer overflow in a library or a misimplemented authentication check in a web framework. It works poorly for vulnerabilities that are emergent properties of how components interact, that depend on the content flowing through a system rather than its code, or that are not attributable to a single vendor's software.

Prompt injection — the dominant attack class against AI agents — illustrates the problem precisely. A prompt injection vulnerability exploits the fact that an AI model cannot reliably distinguish between instructions from its operator and instructions embedded in the data it processes. When an attacker embeds a malicious directive in a document, email, or web page that an agent is asked to summarize, the agent may execute the directive as if it were an authorized instruction. The EchoLeak vulnerability (CVE-2025-32711) realized this scenario against Microsoft 365 Copilot: an attacker who sent a crafted email to a target organization could trigger Copilot to silently exfiltrate sensitive documents to an attacker-controlled endpoint, requiring no further user interaction [3]. Assigning a CVE to EchoLeak was possible because Microsoft is a clearly identified vendor with a well-resourced disclosure process. But the same attack pattern — indirect prompt injection through attacker-controlled content — applies to virtually any AI agent that processes external data, and thousands of community-built agents have no vendor in a position to receive, triage, or remediate a disclosure.

The NVD enrichment pipeline compounds the problem. Even when CVEs are assigned for AI-related vulnerabilities, the NVD's scoring and categorization infrastructure was designed around established CWE categories that map poorly to AI-specific failure modes. Prompt injection has no dedicated CWE entry that captures its full scope. Memory poisoning — where an attacker corrupts an agent's persistent context to influence future behavior — does not fit cleanly into any existing weakness taxonomy. The OWASP Top 10 for Agentic Applications, released in December 2025 following input from over one hundred security researchers and industry practitioners, introduced the risk categories ASI01 through ASI10 specifically because existing taxonomies did not cover the threat surface [5]. That framework — which includes agent goal hijack, tool misuse and exploitation, identity and privilege abuse, agentic supply chain vulnerabilities, and cascading failures — fills an important gap, but it has not yet been integrated into the CVE/NVD pipeline in a way that gives enterprises the same automated visibility they have for conventional software vulnerabilities.

Leadership at the Cybersecurity and Infrastructure Security Agency has publicly acknowledged the gap, suggesting that AI companies "should be better represented" in the CVE program [6]. That framing, while accurate, understates the structural nature of the problem. Better representation by individual vendors does

not solve the attribution problem for community-developed components, the taxonomy problem for AI-specific failure modes, or the reproducibility problem for emergent vulnerabilities. Those require changes to disclosure infrastructure itself.

The Silent Bounty Problem

A vulnerability accepted by a bug bounty program, triaged by a vendor security team, and rewarded with a bounty payment has, in the ordinary course of events, produced a patch and an advisory. In the AI agent ecosystem, that sequence has begun to break down in a pattern researchers have started documenting: reports accepted, bounties paid, no CVE filed, no public advisory published.

The structural incentive for this pattern is straightforward. Bug bounty programs exist partly to discover vulnerabilities before attackers do, but they are also reputation management instruments. A vulnerability in a new AI product category that lacks established CVE precedent is easy to classify as a product limitation rather than a security defect, reducing or eliminating the vendor's formal disclosure obligation. When a vendor's security team accepts a report and pays a bounty without filing a CVE, the researcher receives compensation but the broader security community receives no notification. Enterprises using the affected product have no mechanism for learning that a risk exists. Downstream vendors who depend on the same component cannot patch their own products. The vulnerability effectively disappears from the ecosystem's collective awareness while remaining in production.

The Anthropic SQLite MCP server case illustrates this dynamic. A researcher privately reported a SQL injection vulnerability in Anthropic's reference SQLite MCP server — a component forked more than five thousand times, used as a template for community-built agents, and integrated into production workflows. Anthropic's response was that the repository is an archived demonstration implementation and therefore the vulnerability is "out of scope," with no patch planned [7]. The technical classification may have been accurate; the downstream impact was not limited to that categorization. Developers who forked the reference implementation inherited its vulnerability without knowing it existed, because no CVE was assigned and no advisory was published [7]. The architectural pattern — an archived demo with significant production adoption — is not unique to this case. AI platforms routinely publish reference implementations and sample servers that accrue production deployments faster than their maintainers' intent.

OpenAI's coordinated vulnerability disclosure policy for inbound reports from researchers contains a notable provision: the program does not permit researchers to make public disclosures of vulnerabilities they discover [8]. The policy is not unusual for commercial bug bounty programs, and OpenAI's outbound disclosure policy — governing how it reports vulnerabilities its own systems discover in third-party software — is well-structured and requires human review before release [9]. But the combination of a non-disclosure requirement for inbound researchers and the company's scale of deployment creates an asymmetry:

security researchers who discover significant vulnerabilities in OpenAI's agent infrastructure may have limited recourse if they believe the vendor's remediation is inadequate, because the standard researcher remedy of coordinated disclosure with a deadline is contractually foreclosed.

This is not a criticism unique to any individual vendor. It reflects the immaturity of disclosure norms across an industry that moved from research to wide deployment in compressed timeframes. The absence of community norms specific to AI agent vulnerabilities creates a low-friction path toward opaque remediation, and competitive pressure to avoid publicizing security incidents reinforces that path.

The Multi-Vendor Attribution Maze

Conventional software vulnerability disclosure assumes a reasonably clear answer to the question: who owns this vulnerability? The answer determines who receives the report, who has authority to patch it, and who is responsible for notifying affected users. Agentic AI deployments routinely render that question unanswerable under existing frameworks.

Consider a production AI agent that uses a frontier language model from Provider A, orchestrated through a framework from Provider B, connecting to two MCP servers from community developers C and D, with a system prompt and tool configuration managed by an enterprise deployer E. When a prompt injection through an MCP tool's response data causes the agent to exfiltrate credentials, each stakeholder has a coherent argument that the vulnerability belongs to someone else. Provider A may characterize prompt injection resistance as a property of the model but note that the orchestration layer is responsible for input sanitization. Provider B may characterize the framework's behavior as compliant with the model's API contract. Community developer C may note that the MCP specification does not define security requirements for server responses. The enterprise deployer may have no security team member who understands the technical surface at all.

CSA research on AI agent identity governance has documented the organizational side of this gap: a significant majority of surveyed enterprises report that they cannot reliably distinguish AI agent activity from human activity in their logs, and nearly three-quarters of respondents assessed that their AI agents receive more access than they actually require for their assigned tasks. When accountability for a security event requires reconstructing what an agent did, on whose behalf, with what credentials, and under whose instructions, the absence of robust agent identity and audit infrastructure means that forensic attribution may be impossible even after the fact.

The MCP ecosystem's growth has made this attribution challenge more acute. Between January and February 2026, security researchers published more than thirty CVEs targeting MCP servers, clients, and infrastructure [1]. The CVEs that were assigned went to components with identified maintainers. A

comparable number of exploitable conditions were documented in community implementations where no responsible party was identifiable to receive a disclosure. Dark Reading reported that Microsoft and Anthropic MCP server implementations were found at risk of remote code execution and cloud account takeover [10], representing two of the better-resourced cases in the ecosystem. The long tail of community implementations — where eighty-two percent have file operation vulnerabilities [1] — has no corresponding disclosure infrastructure at all.

The mcp-remote package provides a quantitative illustration of the scale risk. CVE-2025-6514 documented a command injection vulnerability in mcp-remote, a package for connecting to remote MCP servers that had accumulated over four hundred thirty-seven thousand downloads before the vulnerability was reported [1, 11]. The package's purpose — bridging AI model clients to remote MCP servers — placed it in the network path of a substantial fraction of MCP-enabled agent deployments. A single undisclosed vulnerability in that position could affect hundreds of thousands of deployments, but the disclosure process for such a component depends entirely on the maintainer's willingness and capacity to respond to a report from an external researcher.

Case Studies in Agentic Disclosure Failure

The following cases document specific instances where the disclosure process for AI agent vulnerabilities produced outcomes materially worse than the outcomes conventional software disclosure would have generated for equivalent-severity flaws.

Flowise CVE-2025-59528: Seven Months Exposed

Flowise is an open-source, low-code platform for building AI agent workflows and MCP-enabled applications. Its CustomMCP node, which allows users to configure connections to external MCP servers, executes JavaScript code as part of that configuration process without security validation — a code injection condition that grants access to Node.js modules including child_process (for command execution) and the file system [2]. The vulnerability was assigned CVE-2025-59528 with a CVSS base score of 10.0.

The patch was released in Flowise version 3.0.6 in September 2025. In-the-wild exploitation was first confirmed in April 2026, approximately seven months later, with twelve to fifteen thousand publicly reachable Flowise instances still running vulnerable versions [2, 18, 19, 23]. The seven-month gap between patch and confirmed exploitation is not a disclosure failure in the narrow sense — a CVE was assigned and a patch was released. It reflects an ecosystem failure: the user population running vulnerable versions lacked the patch management processes, operational monitoring, or threat intelligence feeds necessary to respond to a maximum-severity vulnerability in a production AI component.

That ecosystem failure is partly a consequence of disclosure norms. Because AI agent platforms have not been historically classified alongside traditional infrastructure components requiring patching cycles, many organizations that deployed Flowise for experimentation or rapid prototyping had no patch management workflow covering it. The emergence of maximum-severity CVEs in AI orchestration platforms has not yet translated into the same operational urgency that a CVSS 10.0 database or operating system vulnerability would generate.

EchoLeak CVE-2025-32711: The Zero-Click Frontier

The EchoLeak vulnerability in Microsoft 365 Copilot established a new category in the AI agent threat landscape: a zero-click attack requiring no user interaction to exfiltrate sensitive organizational data [3]. An attacker who sends a crafted email to a target organization can, without any action by the recipient, trigger Copilot to retrieve and transmit sensitive files to an attacker-controlled endpoint. The attack exploits indirect prompt injection: the email body contains instructions that Copilot interprets as operator directives when processing the message.

Microsoft received the disclosure, assigned CVE-2025-32711, and patched the vulnerability. What the case demonstrates is not a disclosure failure but a threat category failure: the industry's existing mental models for what constitutes a zero-click vulnerability assumed software execution, not AI instruction following. The same architectural pattern — an AI agent that processes external content and acts on it with user-level permissions — is present in thousands of enterprise deployments. The disclosure of EchoLeak produced a patch for one instance of the pattern while leaving the pattern itself unaddressed across the ecosystem.

Langflow CVE-2025-34291: Framework-Level Risk

LangFlow, one of the most widely deployed AI agent orchestration frameworks, was found to contain a remote code execution vulnerability (CVE-2025-34291) that subsequently saw confirmed exploitation in the wild [12]. The vulnerability resided in the application layer of a framework used to build, test, and deploy AI agents — meaning that every application built on LangFlow inherited the exposure. This architecture, where a framework's vulnerabilities propagate to its users, is familiar from conventional software. What distinguishes the AI agent context is the deployment pattern: LangFlow is routinely used by developers without dedicated security resources who are building agents for internal automation, and who may not have the same vulnerability monitoring coverage as production web applications.

The MCP Inspector: Developer Tools as Attack Vectors

The Anthropic MCP Inspector, a developer tool for testing and debugging MCP server implementations, was found to contain a remote code execution and DNS rebinding vulnerability assigned CVE-2025-49596 with a CVSS score of 9.4 [13]. Developer tools occupy an ambiguous position in the vulnerability disclosure

landscape: they are not production infrastructure, but they are installed on the machines of security researchers and developers who have privileged access to production systems. An RCE in the primary debugging tool for MCP server development places the entire MCP development community at risk during precisely the activity — security testing — where their access is broadest.

The vulnerability was reported by Oligo Security and received a CVE assignment, which represents a functioning disclosure outcome. Its significance for this paper is the category it illustrates: the tooling ecosystem around AI agents — the inspectors, simulators, local test servers, and debugging utilities — constitutes a distinct attack surface that conventional software security processes do not routinely cover, but that sits adjacent to some of the most sensitive systems in an AI development organization.

The Compression of Exploitation Windows

The timeline gap between vulnerability disclosure and active exploitation has contracted sharply across the software industry, and that contraction has accelerated since AI-assisted vulnerability analysis became widely available. Analysis of Known Exploited Vulnerabilities in 2025 found that approximately twenty-nine percent showed confirmed exploitation on or before the day their CVE was published [4], up from twenty-four percent the prior year. The mean time from disclosure to confirmed exploitation across all software categories dropped to less than one day in the most severe cases [14].

The mechanism driving that compression is partly AI-enabled offense. AI tools can now analyze a public CVE entry, understand the underlying flaw, and produce a functional proof-of-concept exploit in fifteen minutes at negligible cost [4]. For vulnerabilities in AI agent components — where the attack surface includes both conventional software flaws and AI-specific conditions like prompt injection — the offense side of that equation is well-equipped while the defense side remains organizationally immature. A joint emergency strategy briefing from SANS Institute, CSA, and the OWASP GenAI Security Project in April 2026 warned specifically that AI-driven vulnerability discovery is compressing exploit timelines from weeks to hours, with implications for patch management, threat intelligence, and incident response across the enterprise [14].

This compression has particular consequences for the agentic context. A zero-day in a widely deployed MCP server, a prompt injection pattern in a popular orchestration framework, or a credential exposure condition in an agent identity implementation can be operationalized by a sophisticated attacker before any CVE is published, before any advisory reaches the organizations running vulnerable components, and before any human security analyst has reviewed the disclosure. The seven-month window between the Flowise patch and confirmed exploitation suggests that even when disclosure functions correctly, the subsequent distribution of security awareness to affected users is profoundly inadequate.

The combination of slow organizational response and fast attacker operationalization is not new to security, but the AI agent context adds a third factor: scale of autonomous action. An attacker who gains code execution through a compromised agent does not merely compromise a server — they inherit the agent's credentials, its delegated authorities, and its access to any connected system or data source. The blast radius of a single compromised agent in an enterprise deployment can encompass email, file storage, code repositories, ticketing systems, and external APIs, because that is precisely the integration surface that makes agents valuable. Closing the disclosure gap is therefore not only about reducing the number of unpatched vulnerabilities; it is about limiting the organizational impact when vulnerabilities are inevitably exploited before patches reach all affected deployments.

Bug Bounty Programs Under Structural Strain

AI-generated vulnerability reports have destabilized bug bounty programs in ways that create secondary accountability gaps. The economic model of bug bounty depends on a signal-to-noise ratio that makes triage feasible: a human security researcher who has invested significant effort in finding a vulnerability produces a report with high specificity and reproducibility. When AI tools can generate plausible-sounding vulnerability reports at scale, that model breaks.

In 2025, Daniel Stenberg, the maintainer of the curl networking library, publicly reported that his bug bounty program had become unmanageable after AI-generated reports flooded the queue: fewer than five percent of submitted reports were legitimate, and the effort required to review and debunk the remainder was consuming resources that would otherwise support genuine security work [15]. He ultimately closed the program. The curl case is not isolated; it represents a structural stress on the open-source security ecosystem, where volunteer maintainers cannot absorb the triage burden created by AI-assisted bulk report generation [15].

The consequence relevant to this paper is indirect but significant. When bug bounty programs become economically unviable for open-source AI component maintainers, those maintainers lose their primary channel for receiving vulnerability reports from the external security research community. The choice is between maintaining a program that consumes unsustainable volunteer labor, or operating without any structured external reporting mechanism. Many will choose the latter, creating security visibility gaps for precisely the community-maintained components — MCP servers, agent frameworks, tool integrations — that constitute the longest and least-examined tail of the AI agent attack surface.

Vendors operating large commercial bug bounty programs face a different version of the same problem. AI-assisted bug hunting tools have enabled researchers to submit high volumes of low-quality reports alongside high-quality ones, increasing triage costs even when programs are not overwhelmed with intentional noise. The response from some vendors has been to raise bounty thresholds, add more

restrictive scope definitions, or move from public to invite-only programs — all of which reduce the diversity of the researcher community that examines the vendor's attack surface, which in turn reduces the likelihood of novel vulnerability discovery.

Emerging Frameworks and Remaining Gaps

Several significant frameworks and initiatives have emerged in 2025-2026 that address portions of the disclosure and accountability problem, though none resolves it comprehensively.

The OWASP Top 10 for Agentic Applications, released in December 2025 with input from over one hundred practitioners and peer review by members of the Microsoft AI Red Team, provides the first globally accepted taxonomy of agentic-specific risks [5]. Its ten categories — agent goal hijack, tool misuse and exploitation, identity and privilege abuse, agentic supply chain vulnerabilities, unexpected code execution, memory and context poisoning, insecure inter-agent communication, cascading failures, human-agent trust exploitation, and rogue agents [5] — fill a genuine gap in the threat categorization landscape. The framework is already referenced in Microsoft's Copilot Studio security guidance [22] and AWS security documentation. However, OWASP's framework is a risk taxonomy, not a disclosure process. It defines what classes of vulnerability exist; it does not define how they should be reported, assigned identifiers, or communicated to affected organizations.

MITRE's ATLAS framework has continued to expand its coverage of adversarial AI techniques, reaching sixty-six techniques across fifteen tactics as of October 2025, including fourteen agentic AI attack techniques added in its v5.0 revision [24]. ATLAS provides the closest analog to a structured vulnerability taxonomy for AI-specific attack patterns, but like the OWASP framework, it is oriented toward threat modeling and detection rather than vulnerability disclosure workflows.

On the regulatory side, the European Union's AI Act crossed significant enforcement thresholds in 2025 and 2026. The Act's high-risk AI obligations — which include transparency, robustness, and oversight requirements — take effect for most covered deployments in August 2026, with penalties reaching €35 million or seven percent of global annual turnover for the most serious violations [25]. The Colorado AI Act becomes enforceable in June 2026 [26]. Neither framework directly addresses vulnerability disclosure for AI components, but both create liability exposure that should, in principle, motivate vendors to invest in disclosure processes sufficient to demonstrate regulatory compliance.

The U.S. Cybersecurity and Infrastructure Security Agency published a Request for Information on AI agent security in January 2026, explicitly soliciting input on "security considerations for artificial intelligence agents" including questions about accountability frameworks, disclosure obligations, and minimum security

standards for agentic deployments [16]. The RFI signals regulatory attention to the problem but does not yet constitute regulatory action.

Anthropic's Project Glasswing represents a vendor-initiated attempt to address the adjacent problem of AI-assisted vulnerability discovery in third-party software: the project applies AI models to find vulnerabilities in critical open-source components and discloses them to maintainers under a structured coordinated disclosure framework [17]. For vulnerabilities discovered by AI models, Anthropic's policy requires human review before disclosure and commits to pacing submissions at rates maintainers can absorb [17, 21]. This addresses one dimension of the problem — AI-generated bulk disclosures overwhelming maintainers — but does not address the converse problem of AI platform vendors themselves underreporting vulnerabilities in their own components.

What remains absent is any infrastructure for systematic disclosure across the multi-vendor agentic stack: a shared CVE numbering authority designation for AI-specific components, a community norm establishing accountability for security advisories in community-maintained MCP servers and agent frameworks, and any mechanism for enterprises to receive aggregated AI-agent vulnerability intelligence comparable to the vulnerability feeds they consume for conventional software.

Recommendations

For AI Platform Vendors

AI platform vendors — including model providers, orchestration framework maintainers, and MCP server developers — should treat vulnerability disclosure as a baseline product quality requirement rather than an optional community service. Vendors operating bug bounty programs should review their scope definitions for AI-specific components and ensure that vulnerabilities in reference implementations, sample servers, and developer tools are in scope even if those components are not formally "production" software. The pattern of classifying reference implementations as "archived demos" to avoid disclosure obligations, when those implementations have accumulated significant production adoption, undermines ecosystem trust and leaves deployers without the security information they require.

Vendors who receive vulnerability reports resulting in patches should file CVE assignments systematically rather than selectively. When a patch is released, a CVE should follow. Non-disclosure requirements in researcher-facing bug bounty programs should include explicit sunset provisions: if a vendor has not remediated a disclosed vulnerability within a defined period, the researcher should have the right to proceed with coordinated public disclosure under standard industry timelines.

For the growing category of AI-specific vulnerabilities that do not fit existing CVE scope definitions, vendors should work proactively with MITRE and CISA to develop appropriate taxonomies and assignments rather than allowing ambiguity to serve as a reason for non-disclosure.

For Enterprise AI Deployers

Organizations deploying AI agents in production should apply the same vulnerability management discipline to AI components that they apply to conventional infrastructure. This means maintaining an inventory of every AI component in the stack — model provider, orchestration framework, MCP servers, tool integrations — and establishing a process for monitoring CVE feeds relevant to each component. The current posture of many organizations, in which AI components are treated as black boxes exempt from patching cycles, is inconsistent with the severity of vulnerabilities that have been documented: CVSS 10.0 RCE in agent builders, CVSS 9.6 command injection in MCP infrastructure [11], and zero-click data exfiltration through AI email integration.

Enterprises should also implement agent identity governance practices that enable attribution: each production agent should operate under a distinct, least-privilege identity; its actions should be logged in a format auditable by human reviewers; and the provisioning, credential management, and permission scope of each agent identity should be formally owned by a named team. Organizations where the majority of production agents cannot be distinguished from human activity in the event log are not positioned to respond effectively to agent compromise.

For Security Researchers

Researchers who discover vulnerabilities in AI agent components — particularly community-maintained MCP servers, agent frameworks, and AI tool integrations — should file CVE requests directly through MITRE's web reporting system when a vendor is unresponsive or has declined to file. The CVE program's scope explicitly includes community software without a formal vendor relationship, and researchers should not allow vendor inaction to prevent CVE assignment.

For AI-specific vulnerability classes — prompt injection chains, memory poisoning, agent goal hijack — researchers should document the attack pattern using OWASP's Agentic Top 10 taxonomy alongside or in lieu of traditional CWE classification when submitting CVE requests and publishing advisories. Building a corpus of well-documented AI agent vulnerabilities with consistent taxonomic classification will accelerate the process of integrating AI-specific failure modes into mainstream security tools.

When disclosing prompt injection and other AI-specific vulnerabilities that span multiple vendor components, researchers should notify all vendors whose components contribute to the vulnerable condition rather than selecting a single "responsible" vendor. Multi-party coordinated disclosure is

administratively complex but is the only mechanism that ensures all contributors to the vulnerable state have the opportunity to remediate.

For Standards Bodies and Regulators

MITRE, in coordination with CISA and the broader CVE numbering authority network, should establish a working group with representation from major AI platform vendors, the OWASP GenAI Security Project, and ATLAS maintainers to define scope expansions and taxonomy updates that accommodate AI-specific vulnerability classes. The goal should be a CVE filing process in which a prompt injection with documented cross-organizational impact is as straightforward to file, score, and catalog as a buffer overflow in a C library.

Regulators developing AI security requirements should consider mandating vulnerability disclosure programs for AI products meeting defined deployment thresholds, analogous to disclosure requirements that apply in other critical infrastructure contexts. The EU AI Act and Colorado AI Act create compliance pressure without disclosure specifics; sector-specific guidance that defines minimum disclosure standards would translate that pressure into operational requirements.

CISA's ongoing engagement with AI companies on CVE program participation should be accompanied by concrete technical expectations: AI companies operating at scale should be designated as CVE Numbering Authorities for their own products, should participate in the CNA community's development of AI-specific classification guidance, and should publish annual transparency reports on vulnerability receipt, remediation, and disclosure metrics. Voluntary commitments from major vendors have produced some progress, but the most consequential transparency gaps are in the long tail of community-maintained components where voluntary norms are insufficient.

CSA Resource Alignment

This whitepaper's findings connect directly to several frameworks and programs within the Cloud Security Alliance portfolio. Organizations seeking implementation guidance should treat the following CSA resources as the starting point for remediation planning.

The **CSA AI Controls Matrix (AICM)**, an extension of the Cloud Controls Matrix designed for AI service providers, includes controls addressing security transparency, vulnerability management obligations, and incident disclosure requirements across five role categories: AI customers, application providers, cloud service providers, model providers, and orchestrated service providers. The accountability gap documented in this paper is directly addressed by AICM controls governing disclosure obligations for each provider layer. Enterprises should use the AICM's role-based structure to map disclosure accountability across their agentic stacks and identify where gaps in contractual or operational ownership exist.

The **MAESTRO** threat modeling framework, developed by the CSA AI Safety Initiative, provides a structured methodology for analyzing Multi-Agent Environment, Security, Threat, Risk, and Outcome conditions in autonomous agent deployments. MAESTRO's seven-layer model — from model layer threats through agent layer, data layer, deployment layer, operational layer, trust boundary layer, and organizational context — offers a systematic approach to identifying the vulnerability disclosure responsibilities that correspond to each layer. Organizations conducting threat model reviews of agentic deployments should use MAESTRO to ensure that disclosure accountability is assigned at every layer where a vulnerability could originate.

The **CSA STAR program** (Security, Trust, Assurance, and Risk), and specifically the emerging STAR for AI extension, provides a third-party assurance framework through which AI vendors can demonstrate their security practices — including vulnerability disclosure — to enterprise customers. Procurement teams evaluating AI agent platforms should require STAR for AI attestation or equivalent third-party validation of disclosure practices as a condition of vendor selection. The absence of any public disclosure track record for a platform whose components have documented security vulnerabilities should be treated as a disqualifying characteristic.

CSA's published research on **agentic AI identity governance** and the **2025 Agentic Identity Survey** provides empirical grounding for the agent identity and attribution recommendations in this paper. The survey's finding that a significant majority of enterprises cannot distinguish agent from human activity in their event logs, and that most production agents have more access than their tasks require, directly enables the accountability gap that makes post-exploitation attribution so difficult. Implementing the identity governance controls documented in that research is a prerequisite for meaningful incident response when agentic disclosure failures result in exploitation.

Finally, the CSA's prior research on **bug bounty program design** — examining the conditions under which vulnerability disclosure programs succeed or fail — provides operational guidance for AI platform vendors designing or reforming their disclosure programs. The structural stresses AI-generated report volume places on bug bounty programs are an extension of the organizational challenges that research documented, and the design principles it identified apply directly to AI platform contexts.

Conclusions

The vulnerability disclosure ecosystem that the security industry relies upon is not failing through negligence. It is failing because the assumptions built into its design — clearly bounded software components, identifiable vendor ownership, reproducible flaws, and version-specific patches — do not describe agentic AI systems. Agentic systems are compositional, non-deterministic, persistent, and

delegated in ways that scatter accountability across multiple vendor layers simultaneously, and the community-maintained nature of the MCP and agent framework ecosystems means that many of the most widely deployed components have no organizational capacity to receive and respond to disclosures at all.

The consequences of this structural failure are material. Maximum-severity vulnerabilities in AI agent builders remain actively exploited seven months after patching. Zero-click exfiltration attacks through AI email agents reach production before defenses are generalized. SQL injection in reference implementations propagates through thousands of forks without ever triggering a CVE. The exploitation window across all software is compressing toward zero in categories where AI-enabled offense is active, and the defense side of the agentic stack is nowhere near the corresponding operational maturity.

The recommendations in this paper are not speculative: they describe processes that exist in other software categories and that the industry knows how to implement. The work is coordination work — among vendors, standards bodies, regulators, and security researchers — of the kind that built the existing disclosure ecosystem over three decades. That ecosystem was not built quickly, but the AI agent security surface is growing at a rate that cannot accommodate a three-decade timeline for the corresponding disclosure infrastructure. The organizations, vendors, and regulators who define disclosure norms for AI agents in the next two years will determine the accountability baseline for an ecosystem that may include hundreds of thousands of deployed agent systems within the decade.

References

[1] heyuan110. "MCP Security 2026: 30 CVEs in 60 Days — What Went Wrong." heyuan110.com, March 2026.

[2] The Hacker News. "Flowise AI Agent Builder Under Active CVSS 10.0 RCE Exploitation; 12,000+ Instances Exposed." The Hacker News, April 2026.

[3] Checkmarx. "EchoLeak (CVE-2025-32711): Show Us That AI Security Is Challenging." Checkmarx, 2025.

[4] IntegSec. "In the Age of AI: The Vanishing Gap Between Vulnerability Disclosure and Exploitation." IntegSec Blog, 2025.

[5] OWASP GenAI Security Project. "OWASP Top 10 for Agentic Applications for 2026." OWASP, December 2025.

[6] Infosecurity Magazine. "AI Companies To Play Bigger Role in CVE Program, Says CISA." Infosecurity Magazine, 2025.

[7] Trend Micro. "Why a Classic MCP Server Vulnerability Can Undermine Your Entire AI Agent." Trend Micro Research, 2025.

[8] Bugcrowd. "Bug Bounty: OpenAI." Bugcrowd, 2025.

[9] OpenAI. "Outbound Coordinated Disclosure Policy." OpenAI, 2025.

[10] Dark Reading. "Microsoft & Anthropic MCP Servers at Risk of RCE, Cloud Takeovers." Dark Reading, 2025.

[11] JFrog. "Critical RCE Vulnerability in mcp-remote: CVE-2025-6514 Threatens LLM Clients." JFrog Security Research, 2025.

[12] Obsidian Security. "CVE-2025-34291: Critical Account Takeover and RCE Vulnerability in the Langflow AI Agent & Workflow Platform." Obsidian Security, 2025.

[13] Oligo Security. "Critical RCE Vulnerability in Anthropic MCP Inspector (CVE-2025-49596)." Oligo Security, 2025.

[14] GlobeNewswire. "SANS Institute, Cloud Security Alliance, [un]prompted, and OWASP GenAI Security Project Release Emergency Strategy Briefing." GlobeNewswire, April 14, 2026.

[15] Axios. "AI Bug Hunters Are Reshaping Open-Source Security's Disclosure Programs." Axios, March 2026.

[16] Federal Register. "Request for Information Regarding Security Considerations for Artificial Intelligence Agents." Federal Register, January 8, 2026.

[17] Anthropic. "Project Glasswing: Securing Critical Software for the AI Era." Anthropic, 2025.

[18] BleepingComputer. "Max Severity Flowise RCE Vulnerability Now Exploited in Attacks." BleepingComputer, April 2026.

[19] Security Affairs. "Attackers Exploit Critical Flowise Flaw CVE-2025-59528 for Remote Code Execution." Security Affairs, April 2026.

[20] Red Hat. "MCP Security: The Current Situation." Red Hat Blog, 2025.

[21] Anthropic. "Coordinated Vulnerability Disclosure for Claude-Discovered Vulnerabilities." Anthropic, 2025.

[22] Microsoft Security Blog. "Addressing the OWASP Top 10 Risks in Agentic AI with Microsoft Copilot Studio." Microsoft, March 2026.

[23] CSA Labs. "Flowise CVSS 10.0 RCE: AI Agent Builders Under Attack." Cloud Security Alliance, April 2026.

[24] MITRE. "ATLAS (Adversarial Threat Landscape for AI Systems) — v5.0 Release." MITRE ATLAS, October 2025.

[25] Artificial Intelligence Act EU. "Article 99 — Penalties." artificialintelligenceact.eu, 2024.

[26] Colorado General Assembly. "HB 24-1468: Artificial Intelligence — Consumer Protections." Colorado General Assembly, 2024.