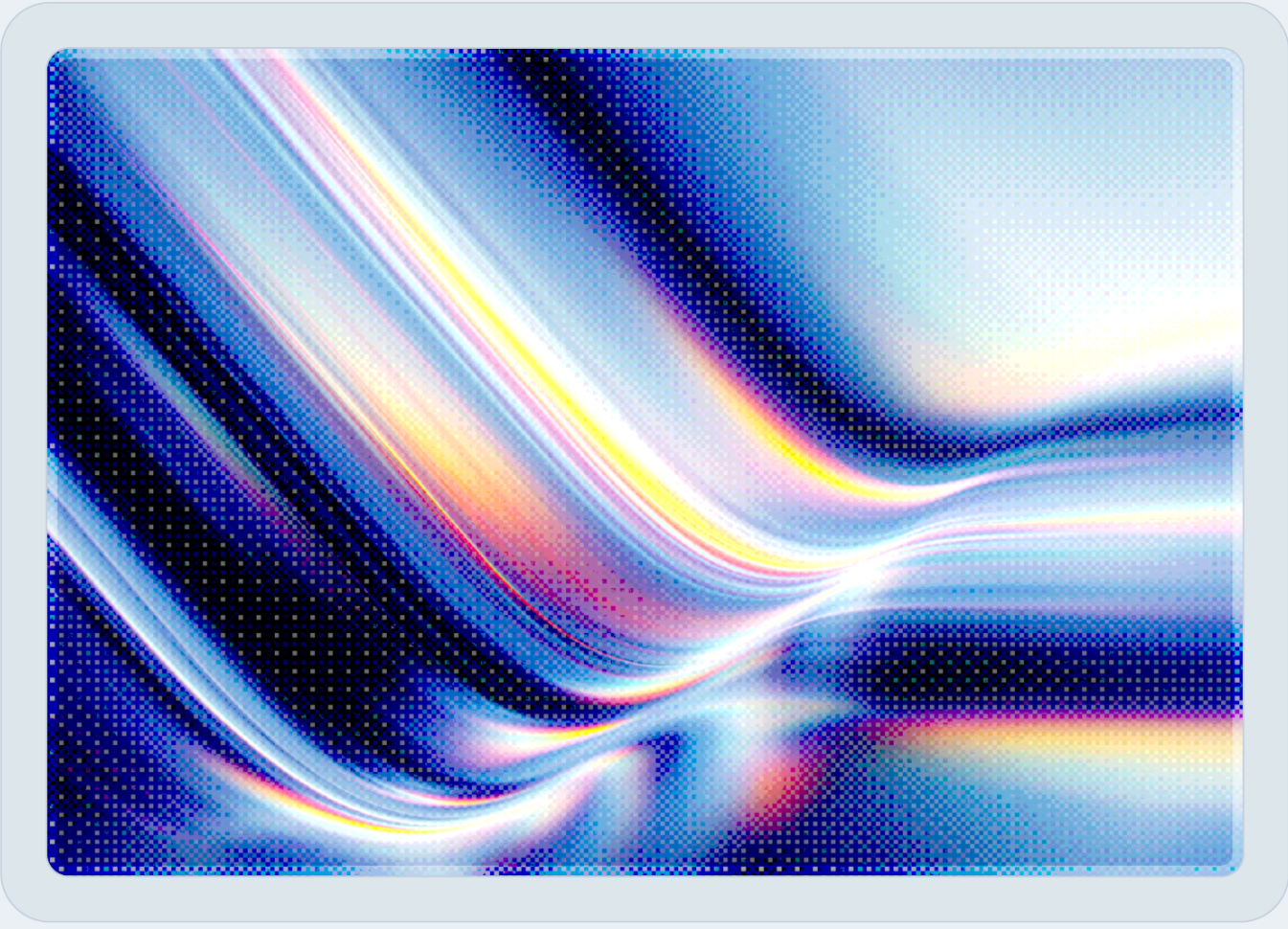
CSAI Foundation | Cloud Security Alliance

The Agentic AI Trust-Boundary Crisis

A Cross-Vendor Pattern of Approval-Gate and Sandbox Failures

2026-08-03

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

Between early and late July 2026, security researchers published five independent vulnerability disclosures against five unrelated agentic AI products: AWS's Kiro coding IDE, Microsoft's Azure DevOps Model Context Protocol (MCP) server, a cluster of open-source Android mobile-agent frameworks, Anthropic's Claude Cowork desktop agent, and OpenAI's ChatGPT Agent Builder. No two disclosures came from the same research team, and no two affected products share a codebase, a vendor, or even an operating system. Yet every one of the five defeats the same class of control. Two of the five (AWS Kiro and ChatGPT Agent Builder) show that a human-approval prompt can be technically satisfied while authorizing something entirely different from what the human believes they approved — a pattern that a related, non-primary disclosure, the symlink-based GhostApproval finding from Wiz Research, generalizes across an entire product category. Two more of the five (Claude Cowork and the Android agent frameworks) show that an execution sandbox marketed as the safety boundary between an untrusted agent and a trusted host can be walked straight through, using nothing more exotic than a kernel bug or an unescaped shell command. A separate thread, cutting across the Azure DevOps case and the ChatGPT case already described above, shows that even where an approval step functions exactly as designed, the authority it grants can be silently redirected to targets the human never contemplated.

This paper argues that these are not five unrelated bugs that happened to cluster in July. They are five expressions of one underlying design gap: agentic AI systems consistently treat the *appearance* of a safe boundary — a confirmation dialog, a virtual machine, a scoped credential — as equivalent to an *enforced* one, without verifying that what the human sees, what the sandbox contains, and what the credential actually authorizes are the same thing the system claims they are. The Cloud Security Alliance's own research program has been documenting fragments of this pattern since at least March 2026, under headings including confused-deputy attacks [16], symlink-based trust-boundary gaps [15], and sandbox trust-handoff flaws [14]. This paper's contribution is to place the July 2026 disclosures side by side, show that they are instances of a common structural failure rather than isolated vendor defects, and translate that finding into concrete guidance for security leaders who must decide, this quarter, how much autonomy to grant the agentic AI systems already running in their environments.

Introduction and Background

Agentic AI — large language model-driven systems that plan multi-step tasks, invoke tools, and act on external systems with limited human supervision — moved from developer preview to production deployment across the software industry through 2025 and into 2026. Coding assistants such as AWS Kiro,

Anthropic's Claude Code and Claude Cowork, Cursor, and GitHub Copilot's agent mode now routinely execute shell commands, edit files, and call external services on a developer's behalf. Enterprise platforms such as Microsoft's Azure DevOps and OpenAI's ChatGPT Workspace have layered agentic capability on top of existing collaboration tools, letting an agent read pull requests, trigger pipelines, or operate connected mailboxes and chat channels with the permissions of the human who invoked it. The Model Context Protocol (MCP), introduced by Anthropic in late 2024, has become the dominant integration layer connecting these agents to the systems they act on, and CSA's own research has tracked more than thirty CVEs against MCP implementations in the first two months of 2026 alone [19].

This rapid expansion in capability has been accompanied, almost without exception, by the same two safety arguments. The first is the human-approval gate: before an agent takes a consequential action — writing outside a project folder, merging code, sending a message, granting a connector — the interface pauses and asks a person to confirm. The second is the execution sandbox: the agent's actual code execution happens inside an isolated environment, a container or a virtual machine, so that even if the model is manipulated into attempting something malicious, the blast radius is contained to a disposable compartment rather than the host system. Vendors present both controls as load-bearing. Regulators and standards bodies, including the United States National Security Agency in its May 2026 Cybersecurity Information Sheet on MCP security [20], have pointed to human-in-the-loop review and sandboxed execution as baseline expectations for any organization deploying agentic AI.

The five incidents examined in this paper were disclosed within roughly four weeks of one another, between July 8 and July 27, 2026, by five different research teams working independently: Intezer and Kodem Security (AWS Kiro), Manifold Security (Azure DevOps MCP server), an academic team publishing on arXiv (the Android agent frameworks), Accomplish AI (Claude Cowork), and Zenity Labs (ChatGPT Agent Builder). None of the five researchers cite the others' work, and nothing in the public record suggests coordinated disclosure. That independence is precisely what makes the pattern significant: five teams, working on five different platforms with five different research methodologies, converged on the same underlying finding — that the approval gate or the sandbox boundary they were testing looked solid from the outside and was hollow underneath.

The Five-Incident Corpus

The table below summarizes the five disclosures by vendor, mechanism, and disposition as of this writing. The narrative sections that follow group them by the specific control each one defeats, since the grouping — rather than the vendor identity — is what carries the analytical weight of this paper.

Product / Vendor	Discovering Researchers	Control Defeated	Technical Mechanism	Fix Status (as of late July 2026)
AWS Kiro	Intezer, Kodem Security	Approval gate	Hidden web-page text redirects an approved "fetch a URL" action into an unreviewed rewrite of Kiro's MCP configuration file, which Kiro then executes with developer privileges [1] [2]	Fixed in Kiro 0.11; CVE-2026-10591 assigned, CVSS 8.8 [3]
Azure DevOps MCP server	Manifold Security	Delegated authority (confused deputy)	HTML comments in a pull request description are invisible in the web UI but returned verbatim by the API, letting an attacker's hidden instructions execute with the reviewing developer's cross-project credentials [4][5]	Unpatched; no CVE assigned as of disclosure [4][5]
AppAgent, AppAgentX, Mobile-Agent-v3, Open-AutoGLM, MobA (Android agent frameworks)	Academic researchers (arXiv)	Sandbox / host boundary	Vision-language models read invisible or tampered on-screen text and relay it into unescaped shell commands, achieving code execution on the host PC controlling the phone [6][7]	Unpatched on all five projects' main branches as of disclosure; no CVEs assigned [6] [7]
Claude Cowork	Accomplish AI	Sandbox / VM boundary	A Linux kernel privilege-escalation bug, exploitable because the host filesystem is	Closed by Anthropic as "informative"; latest release

Product / Vendor	Discovering Researchers	Control Defeated	Technical Mechanism	Fix Status (as of late July 2026)
			mounted read-write inside the agent's VM, lets the agent obtain guest-root and reach host files [8][9]	defaults to cloud execution rather than the affected local mode [8][9]
ChatGPT Agent Builder	Zenity Labs	Approval gate (via CSRF)	A crafted URL auto-submits agent-creation parameters, standing up a persistent, fully-connected rogue agent the instant a logged-in victim clicks the link — no approval screen is ever shown	Fixed by OpenAI, June 8, 2026 [10][11]

Two additional, closely related disclosures inform the analysis without being counted among the five primary incidents, because they were reported by third parties rather than named in this paper's original research brief. The boundary between the five and these two is accordingly an editorial one, reflecting the scope of that brief, rather than a claim that only five such incidents exist; readers should treat the additional disclosures below as reinforcing evidence for the pattern rather than as data points that were tested against the pattern and happened to fit. Wiz Research's July 8, 2026 disclosure of "GhostApproval" found that six widely used AI coding assistants — including Amazon Q Developer, Claude Code, Cursor, Windsurf, Augment, and Google Antigravity — could each be tricked by a symlink inside a malicious repository into writing attacker-controlled content, such as an SSH `authorized_keys` entry, to a location entirely different from the one displayed in the tool's approval dialog [12][13]. Separately, the Cloud Security Alliance's own July 22, 2026 research on AI coding agent sandbox escapes, building on findings attributed to Pillar Security, found that Cursor, Codex, Gemini CLI, and Google Antigravity share a structural sandbox flaw in which an agent that stays entirely within its permitted sandbox can still write a file that a *trusted host-level tool outside the sandbox* later executes, loads, or scans — producing an escape without the agent ever technically leaving its box [14]. Both findings reinforce, from independent angles, the pattern this paper traces through its five named incidents.

Pattern One: Approval Gates That Approve the Wrong Thing

The approval dialog is the most visible safety control in agentic AI products, and it is also the control most consistently defeated by a mismatch between what a human is shown and what a system actually does. The AWS Kiro disclosure is the clearest illustration. Kiro's protections were built around the assumption that any action requiring elevated trust — writing to a sensitive file, running a shell command — would surface a prompt the developer could approve or deny. Researchers at Intezer and Kodem Security demonstrated that this assumption held for the action Kiro's interface actually displayed, while failing entirely for the action that action set in motion. A developer who asked Kiro to summarize a web page was, in the researchers' words, approving nothing more than "fetching a URL." But the page contained hidden text — rendered in a color matching its background, invisible to a human scanning the page — instructing Kiro to rewrite its own MCP server configuration file, `~/.kiro/settings/mcp.json`. Kiro automatically reloads that file when it changes and launches whatever command it specifies, with the developer's own privileges and with no additional approval step, because from the tool's perspective the file write and the subsequent execution were simply consequences of the URL fetch the developer had already blessed [1][2]. AWS's own summary of the fix, published as CVE-2026-10591 with a CVSS score of 8.8, confirms the root cause as "insufficient access control restrictions" on writes to execution-sensitive paths, and the remedy — a new "protected paths" list requiring explicit approval before any write to files like `mcp.json` — is a direct acknowledgment that the original approval boundary was drawn around the wrong action [3].

The ChatGPT Agent Builder flaw that Zenity Labs named "AgentForger" achieves a structurally similar outcome by a different route: it does not trick an approval dialog into misrepresenting its target, it simply ensures no dialog appears at all. OpenAI's Agent Builder tool accepts initialization parameters, including an agent template and a natural-language starting prompt, as ordinary URL query parameters. Zenity's researchers found that when a user with an active ChatGPT session and at least one connected integration — Outlook, Gmail, Slack, or Teams — opened a specially crafted link, the `initial_assistant_prompt` parameter was not merely pre-filled into a text box for the user to review and submit. It was submitted automatically, on page load, executing the embedded instructions before the victim had any opportunity to see, let alone approve, what was about to happen. The forged agent that resulted was granted every available connector with approval settings set to "never ask," scheduled to run every five minutes, and launched immediately — establishing what Zenity termed a persistent "agentic insider" capable of harvesting documents, credentials, and messages, and of impersonating the victim to third parties. This is a cross-site request forgery attack in its mechanics, but its consequence is the same as Kiro's: an action that the product's own design assumes requires a human decision is instead executed with no human decision made at all [10][11].

Wiz Research's GhostApproval finding, while outside this paper's five primary disclosures, generalizes the Kiro pattern across an entire product category and gives it a name worth adopting industry-wide: the *approval-then-resolve* failure. Each of the six affected coding assistants displayed a file path to the developer, obtained the developer's approval for an operation on that path, and only then resolved the path through the filesystem — by which point a symbolic link planted inside a cloned repository could silently redirect the write to a sensitive location such as a shell startup script or an SSH key file, entirely outside the visible project workspace. As CSA's own analysis of the disclosure observes [15], this is decades-old Unix symlink-following behavior (CWE-61) combined with a newer failure mode, user-interface misrepresentation of the actual write target (CWE-451); the combination matters specifically because it defeats a *human-in-the-loop* control that organizations have adopted precisely because they distrust the AI system's judgment on its own. When the loop itself is fed false information, the human's approval no longer functions as a safeguard against the action actually taken [12][13].

Pattern Two: Confused Deputies and Delegated Authority

A second cluster of failures does not depend on deceiving a human at all. It depends on an agent correctly using the authority it was actually and legitimately granted — just on behalf of someone other than the person who granted it. This is the classical confused-deputy problem, first described in the access-control literature decades before agentic AI existed, and it recurs because agentic systems inherit a property that makes the problem sharper than it has ever been: they act with a human's full, standing authority in response to *any* content they process, including content written by people who have no relationship to that human at all.

Manifold Security's disclosure against Microsoft's Azure DevOps MCP server is the clearest case in the July 2026 corpus. Microsoft's server exposes tools that let a connected AI agent read pull requests, inspect pipelines, browse wikis, and post comments, all under the authenticated identity of the developer who invoked the agent — a design that lets a reviewer ask an agent to "summarize this PR" without the agent needing any separate credential of its own. Markdown, the format Azure DevOps uses for PR descriptions, supports HTML comments that render invisibly in a browser; Microsoft's REST API, by contrast, returns those comments verbatim as part of the description text. Manifold found that while Microsoft had already applied a defense called "spotlighting" — wrapping untrusted content in explicit delimiters that instruct the model to treat it as data rather than commands — to several tools that ingest external content, the specific tool that retrieves pull request descriptions, `repo_get_pull_request_by_id`, was not among them. An attacker needs only contributor-level access to a single project to open a pull request whose description hides an instruction telling the reviewer's agent to approve the change, trigger a pipeline in a completely separate project the attacker cannot reach directly, retrieve a confidential wiki page tied to that

pipeline, and post its contents back as a comment the attacker can read — all while instructing the agent to say nothing about it to the human reviewer. Because the agent acts with the reviewing developer's own token, every one of those steps succeeds using authority the attacker could never have exercised independently. Microsoft's public response characterized the finding as "a known class of AI risk" and recommended limiting project access and reviewing changes before invoking AI tools, guidance that, in this paper's assessment, does not address the actual defect, since the entire premise of the attack is that the malicious content is, by design, invisible to a human scanning the pull request [4][5].

The ChatGPT Agent Builder case belongs equally to this pattern once the CSRF mechanics are set aside: the forged agent that AgentForger creates does not need to steal a credential, because it inherits, by design, the full set of connector permissions and organizational access already available to the victim who unknowingly spawned it. And CSA's own March 2026 research on confused-deputy attacks against autonomous AI agents [16], published well before either the Azure DevOps or ChatGPT disclosures, had already formalized the underlying chain: an adversary introduces malicious instructions into content the agent will process without needing to authenticate as anyone; the agent inherits and acts on those instructions using its own legitimate, standing credentials; the resulting actions propagate across systems the agent is authorized to touch; and, in multi-agent architectures, a compromised agent's output can itself carry a fresh injection into whatever system consumes it next, continuing the chain without further human involvement at any stage. Both July disclosures fit this framework exactly, six months after CSA first described it in the abstract, which is itself evidence that the pattern is structural rather than incidental to any one vendor's implementation choices.

Pattern Three: Sandboxes That Assume a Trusted Boundary

The third failure mode abandons the approval gate entirely and instead targets the control organizations rely on when an agent's own judgment cannot be trusted at all: physical or virtual isolation between the agent's execution environment and the systems an attacker actually wants to reach. Two disclosures in the July 2026 corpus show that this isolation, in current commercial and open-source implementations, does not hold to the extent vendors' descriptions imply.

Accomplish AI's disclosure against Claude Cowork, which the researchers named "SharedRoot," is the more consequential of the two because of its scale: the affected local-execution mode reportedly covered approximately 500,000 macOS users at the time of disclosure. Claude Cowork runs agent sessions inside a Linux virtual machine created through Apple's Virtualization framework, with an unprivileged user account and seccomp filtering layered on top — a defense-in-depth design that looks, on paper, like exactly the kind of sandboxing organizations have been told to expect. Accomplish AI found that the entire host Mac

filesystem was mounted read-write into that VM, accessible to any process running as guest-root inside it. The researchers then chained this design decision to CVE-2026-46331, a Linux kernel privilege-escalation flaw in the `act_pedit` traffic-control packet-editing subsystem, exploitable by loading the affected kernel module from within an unprivileged user namespace. The combination let an agent process inside the VM obtain guest-root and, from there, read and write across the entire host filesystem — including SSH private keys, cloud credentials, and browser data belonging to the logged-in Mac user — "with no permission prompt anywhere," in the researchers' description. Anthropic's response, closing the report as "informative" without shipping a fix, and instead defaulting the product to cloud-based execution going forward, leaves users who deliberately choose local execution exposed to the same underlying design [8][9].

The academic disclosure against five open-source Android agent frameworks — AppAgent, AppAgentX, Mobile-Agent-v3, Open-AutoGLM, and MobA — demonstrates the same category of failure with a far lower barrier to exploitation, because it requires no kernel vulnerability at all. Each framework pairs a vision-language model, which interprets what it sees on a phone's screen, with a host-side orchestrator that translates the model's decisions into Android Debug Bridge (ADB) commands issued to the device. The researchers first showed that every model they tested — including GPT-4o, Claude Opus 4.5, Gemini 3 Pro, and GLM-4V — reliably read text rendered at just 2% opacity, invisible to a human glancing at the same screen, succeeding in at least eighteen of twenty trials per model. They then showed that several of the frameworks wrote screenshots to predictable, fixed device locations with observable timing windows between capture and retrieval — as long as 500 milliseconds in some cases — during which a background process polling every five to ten milliseconds could intercept and silently alter the image the agent would go on to interpret. The decisive step came last: when the tampered or invisible text reached the host orchestrator, it was concatenated directly into a shell command and executed with `shell=True`, with only spaces and single quotes stripped from the input. A payload as simple as `test;pwd>rce_success` was enough to demonstrate code execution on the host PC against AppAgent, and a comparable payload that launched a calculator application succeeded in twenty out of twenty trials across four of the five frameworks. As of the researchers' disclosure, no CVE had been assigned to any of the five projects, none of which maintain a published vulnerability-disclosure policy, and the vulnerable code remained on every project's main branch [6][7].

What unites the Claude Cowork and Android findings is not the specific vulnerability class — a kernel privilege-escalation bug bears little technical resemblance to an unescaped shell string — but the design assumption both defeat. Each system's threat model implicitly treated the boundary between "agent" and "trusted execution surface" as fixed and enforced by construction, when in each case that boundary depended on an additional, unstated assumption: that the filesystem shared into the sandbox was appropriately scoped, or that text read from a screen could be trusted to be exactly what it appeared to be. CSA's July 2026 research on sandbox trust-handoff flaws, examining a separate but structurally identical pattern across Cursor, Codex, Gemini CLI, and Google Antigravity, reaches the same conclusion from yet another angle: an agent can honor every rule of its own sandbox and still achieve an escape, because it only has to write a file that a *trusted tool outside the sandbox* will later execute, load, or scan on its behalf.

Isolation that stops at the boundary of the agent's own process, rather than extending to every system that will subsequently act on the agent's output, is not isolation at all in any threat model that includes attacker-controlled input [14].

Why the Pattern Recurs: A Structural Diagnosis

This paper's analysis suggests that five vendors did not independently make the same coding mistake in the same month so much as they made, independently, the same architectural assumption in five different products — an assumption specific enough to state plainly: each system treats a *representation* of a safety boundary — the text in an approval dialog, the walls of a virtual machine, the scope implied by a credential — as though it were, by construction, identical to the boundary itself. In every one of the five incidents examined here, an attacker's actual leverage came not from breaking cryptography or bypassing authentication, but from exploiting the gap between what a control claims to constrain and what it verifiably does constrain.

This gap recurs for three structural reasons that are common across the industry rather than specific to any one product. First, large language models process instructions and data in the same channel. An agent that reads a web page, a pull request description, or a phone screen has no architecturally guaranteed way to distinguish an instruction the operator intended from text an attacker planted in content the operator asked the agent to process — the model must be told, explicitly and consistently, which parts of its input are trustworthy, and as the Azure DevOps case shows, "consistently" is the operative word that most implementations still fail to satisfy even within a single product. Second, approval and isolation controls are typically implemented as a single check performed once, at the point where an action is initiated, rather than as an invariant enforced continuously through to the point where the action actually executes. Kiro's approval covered the URL fetch but not the configuration write it triggered; GhostApproval's dialogs covered the displayed path but not the resolved one; the Android frameworks' models read the screen once but the screen's content could change in the window before the command executed. Third, the industry currently has no standard, auditable way for an organization to verify, before adopting a product, that its approval gates and sandbox boundaries have been applied *uniformly* across every code path that touches untrusted content, rather than selectively across the paths a vendor's own team happened to consider highest-risk. Microsoft's own defense against exactly this class of attack — spotlighting — already existed in the Azure DevOps MCP server before Manifold's disclosure; it simply had not been applied to one tool among several. That is not evidence of a careless vendor. This instance suggests that partial, inconsistent coverage of a known mitigation may be a common outcome when no external process forces complete coverage — a hypothesis the Manifold case supports but that the other four incidents in this corpus, which involve different failure mechanisms entirely, do not independently confirm.

None of the five vendors examined here disputes the underlying facts of the vulnerability reported against their product, with the partial exception of Anthropic's disposition of the Claude Cowork report. That absence of dispute is worth noting: it suggests these are not contested edge cases at the margins of what agentic AI is supposed to do, but confirmed gaps in controls each vendor already claims to provide.

Recommendations

Immediate Actions

Security teams operating any agentic AI product against enterprise systems should inventory which of that product's tools or capabilities process content authored by someone other than the person who invoked the agent — pull request descriptions, web pages, screenshots, incoming messages — since every incident in this corpus traces back to exactly that kind of content. Where a product cannot demonstrate that untrusted-content handling is applied consistently across its full tool catalog, teams should disable the higher-risk tools in that catalog (cross-project actions, file writes outside an explicit workspace, connector access with auto-approval enabled) rather than assuming a vendor's general security posture extends uniformly to every feature. Teams running AI coding agents locally, including Claude Cowork and any comparable desktop agent that offers a local-execution mode, should default to cloud or remote execution where the vendor supports it, and should audit local configurations for filesystem shares broader than the specific project directory the agent needs.

Short-Term Mitigations

Organizations should scope every credential, token, or connector grant issued to an agent as narrowly as the underlying workflow permits, limiting reach to a single project, repository, or mailbox rather than an organization-wide grant, so that a successful injection cannot pivot beyond the immediate context the way the Azure DevOps and AgentForger proof-of-concepts both did. Approval workflows should be redesigned so that the action a human is shown and the action the system will actually execute are verified to be the same action at the point of execution, not merely at the point of initial confirmation — a control that would have interrupted the Kiro, GhostApproval, and Cowork chains at their pivot points. Where an agent's underlying model reads external, potentially attacker-influenced input as part of a task — a web page, a screenshot, an API response — that input should be passed through explicit content-provenance handling (comparable to the "spotlighting" defense Microsoft applies unevenly today) so the model can distinguish instructions the operator issued from data the operator merely asked the agent to summarize.

Strategic Considerations

Security leaders should treat "agent acting with human credentials" as a distinct privilege tier requiring its own least-privilege architecture, separate from the identity model built for human users, rather than assuming an agent inherits appropriate scoping simply by inheriting a human's token. Procurement and vendor-risk processes for any agentic AI product should require the vendor to demonstrate, not merely assert, that approval-gate and sandbox controls are applied uniformly across every tool and code path that processes external content, and should treat partial or inconsistent coverage as a material finding rather than an acceptable interim state. Finally, organizations should expect this pattern to recur across additional vendors and products through the remainder of 2026, given that five independent research teams found materially identical structural gaps in five unrelated products within a single month; a security program that reacts to each disclosure individually, rather than auditing its own agentic AI deployments against the underlying pattern described in this paper, will likely be addressing the sixth and seventh instances of the same defect well after they have already been exploited elsewhere.

CSA Resource Alignment

This paper's central claim — that five independently discovered vulnerabilities are expressions of one structural gap rather than five unrelated defects — builds directly on a body of CSA research published across the first seven months of 2026, and organizations acting on this paper's recommendations should treat that research as the operational detail behind the pattern described here rather than as background reading.

CSA's Confused Deputy Attacks on Autonomous AI Agents research note [16], published in March 2026, formalized the four-stage chain — injection vector, authority inheritance, action propagation, authority re-delegation — that this paper applies to both the Azure DevOps and ChatGPT Agent Builder incidents. That the framework, developed months before either disclosure, maps onto both without modification is itself evidence that the underlying defect is structural rather than product-specific, and organizations should use that note's admission-control and behavioral-anomaly-detection recommendations as the concrete starting point for the delegated-authority mitigations this paper recommends in the short term.

CSA's July 22, 2026 research on AI coding agent sandbox escapes documents the same "trusted tool outside the sandbox" failure this paper traces through the Claude Cowork and Android agent-framework disclosures, finding that Cursor, Codex, Gemini CLI, and Google Antigravity share a structural pattern in which an agent that never technically leaves its sandbox can still cause a trusted external tool to execute, load, or scan a file the agent wrote — extending, by the CSA analysis, to GitHub Copilot Agent and Claude

Code as well. Security architects evaluating sandbox controls for any coding agent should treat that note's isolation-boundary analysis and its recommendations on least-privilege MCP and coding-agent architecture [14] as the reference architecture against which this paper's sandbox-related findings should be evaluated.

The approval-gate deception pattern this paper documents in the AWS Kiro and ChatGPT Agent Builder cases parallels, at the level of root cause, the GhostApproval finding CSA analyzed in July 2026 [15]: a human-in-the-loop control is only as trustworthy as the information presented within that loop, and once the interface and the underlying execution path diverge, human approval becomes, in that note's framing, "a formality" rather than a safeguard. Organizations that have adopted human-approval gates specifically because they distrust agentic AI's autonomous judgment should read that finding as a direct challenge to the assumption that the gate itself is trustworthy by default.

Finally, CSA's own August 1, 2026 research note, Invisible PR Comments Hijack AI Code Review Agents [17], provides the most granular treatment available of the Azure DevOps incident this paper discusses at the pattern level, including a detailed breakdown of which of Microsoft's MCP tools apply prompt-injection defenses and which do not. Readers seeking incident-specific remediation guidance for that disclosure, beyond the cross-vendor recommendations offered here, should consult that note directly. Across all four of these CSA resources, and across this paper's own recommendations, the AI Controls Matrix (AICM) v1.1 [18] — particularly its Identity and Access Management and Application and Interface Security domains — functions as the common control baseline against which the least-privilege scoping, content-provenance handling, and uniform-coverage requirements recommended above should be assessed.

References

[1] The Hacker News. "AWS Kiro Flaw Let a Poisoned Web Page Rewrite Its Config and Run Code." The Hacker News, July 2026.

[2] Kodem Security. "AWS Kiro RCE: Prompt Injection to Code Execution." Kodem Security, 2026.

[3] Amazon Web Services. "AWS Security Bulletin AWS-2026-037: CVE-2026-10591." AWS, 2026.

[4] The Hacker News. "Microsoft Azure DevOps MCP Flaw Lets Hidden PR Comments Hijack AI Review Agents." The Hacker News, July 2026.

[5] Manifold Security. "When Your AI Reviewer Works for the Attacker: A Confused-Deputy Bug in Microsoft's Azure DevOps MCP Server." Manifold Security, July 2026.

[6] The Hacker News. "Open-Source Android AI Agents Could Let Invisible Screen Text Run Code on Host PCs." The Hacker News, July 2026.

[7] Zhang, Zidong, et al. "(A)I Sees What You Don't: Exploiting New Attack Surfaces in Third-Party Mobile Agents." arXiv, July 2026.

[8] The Hacker News. "Claude Cowork Flaw Could Let AI Agent Escape Its VM and Access Mac Files." The Hacker News, July 2026.

[9] Accomplish AI. "SharedRoot: Escaping the Claude Cowork Sandbox." Accomplish AI, July 2026.

[10] The Hacker News. "ChatGPT AgentForger Flaw Could Deploy Rogue Workspace Agents via a Phishing Link." The Hacker News, July 2026.

[11] Zenity Labs. "AgentForger, Part 1: ChatGPT Cross-Site Agent Forgery." Zenity Labs, July 2026.

[12] The Hacker News. "GhostApproval Symlink Flaws Could Let Malicious Repos Run Code in AI Coding Agents." The Hacker News, July 2026.

[13] Wiz Research. "GhostApproval: A Trust Boundary Gap in AI Coding Assistants." Wiz, July 2026.

[14] Cloud Security Alliance. "AI Coding Agent Sandbox Escapes: The Trust Handoff Flaw." Cloud Security Alliance, July 2026.

[15] Cloud Security Alliance. "GhostApproval: Symlink Trust Gap in AI Coding Assistants." Cloud Security Alliance, July 2026.

[16] Cloud Security Alliance. "Confused Deputy Attacks on Autonomous AI Agents." Cloud Security Alliance, March 2026.

[17] Cloud Security Alliance. "Invisible PR Comments Hijack AI Code Review Agents." Cloud Security Alliance, August 2026.

[18] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2026.

[19] Cloud Security Alliance. "CVE and CWE Agentic Vulnerability Catalog: Weakness Classes Introduced by Autonomous AI Agents." Cloud Security Alliance, March 2026.

[20] National Security Agency. "Model Context Protocol (MCP): Security Design Considerations for AI-Driven Automation." NSA Cybersecurity Information Sheet, May 2026.

