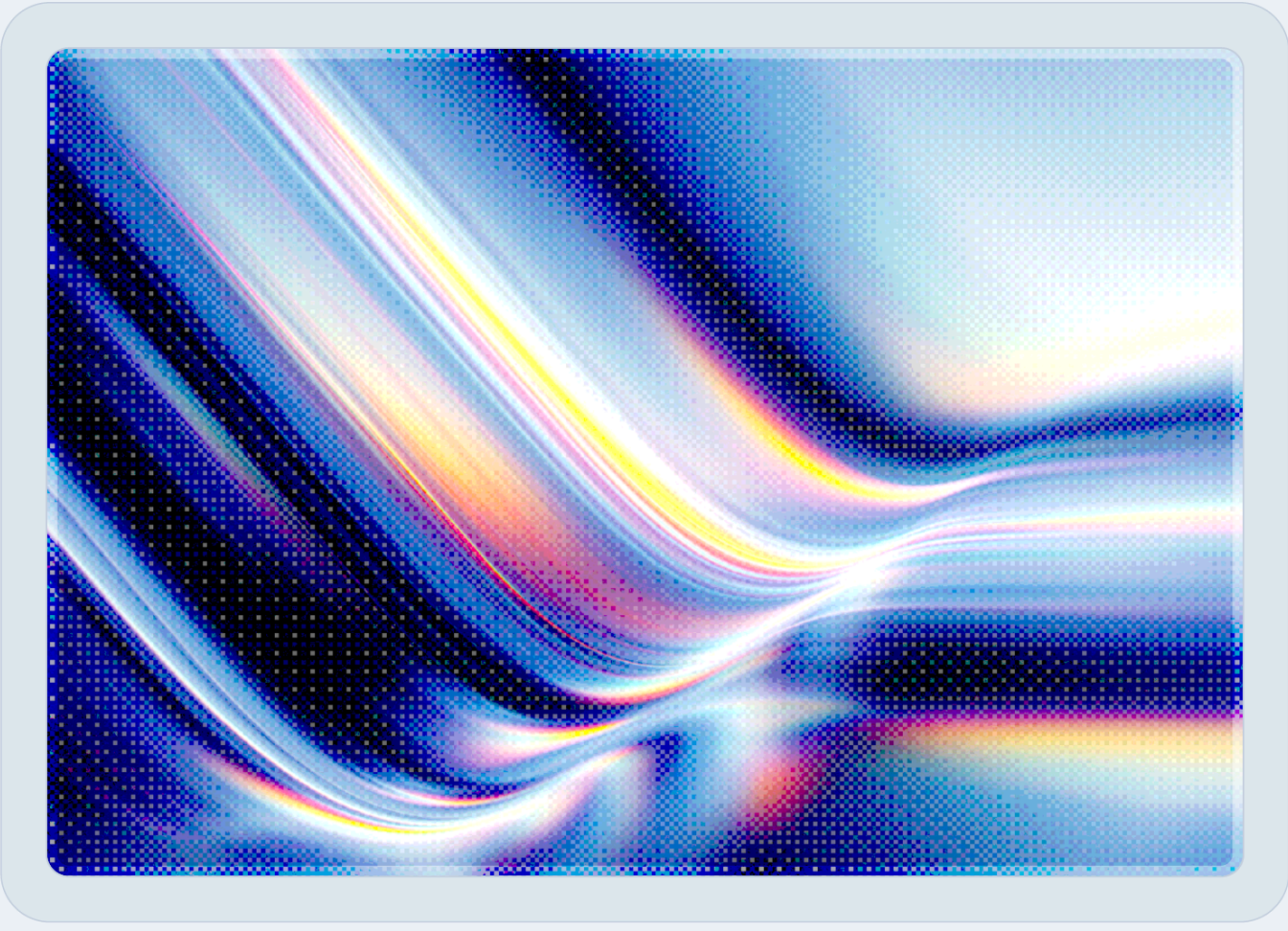
CSAI Foundation | Cloud Security Alliance

Gemini CLI CVSS 10.0: Pre-Sandbox RCE in CI/CD Agents

Workspace Trust Bypass Enables Configuration Injection Before Execution Isolation

2026-05-01

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- Google's Gemini CLI received a CVSS 10.0 (maximum severity) rating for **GHSA-wpqr-6v78-jr5g**, a remote code execution vulnerability disclosed on April 29–30, 2026, affecting headless deployments of the CLI in CI/CD pipelines.
- The root cause is a workspace trust bypass: in headless mode, Gemini CLI automatically trusted the current workspace folder and loaded any configuration found in `.gemini/` directories without user review, sandboxing, or approval—before the agent's execution sandbox ever initialized.
- A secondary flaw allowed `--yolo` execution mode to bypass configured tool allowlists, permitting arbitrary shell commands regardless of restrictions defined in `settings.json`.
- Successful exploitation gives an unprivileged external attacker—one who can open a pull request—access to the full execution privileges of the CI workflow: secrets, credentials, source code, and downstream deployment pipelines.
- Patches are available in `@google/gemini-cli` 0.39.1 (and 0.40.0-preview.3) and `google-github-actions/run-gemini-cli` 0.1.22. Organizations should update immediately and audit any workflow that processes untrusted external content.
- This vulnerability is a concrete illustration of MAESTRO Layer 6 (Deployment Infrastructure) and Layer 7 (Agent Ecosystem) risks: AI coding agents embedded in CI/CD pipelines inherit the execution context of a trusted contributor, making their configuration attack surface indistinguishable from the pipeline itself.

Background

AI coding agents have moved from developer workstations into automated CI/CD pipelines. Tools like Google's Gemini CLI—open-source since mid-2025—are increasingly deployed as GitHub Actions to review pull requests, triage issues, generate code, and run automated workflows that would otherwise require human attention. In this configuration, the agent operates in headless, non-interactive mode: there is no human in the loop to review or approve tool invocations, and the agent's execution context carries whatever secrets and permissions the workflow holds.

This operational pattern creates a substantially different threat model than interactive use on a developer's local machine. When a developer runs an AI CLI tool interactively, the tool can present trust prompts, the user can review proposed commands, and the developer's own judgment serves as a last-resort control. In a CI/CD pipeline, none of those properties hold. The agent runs unattended, trusted inputs are mixed with untrusted external content—pull request bodies, issue comments, repository files submitted by outside contributors—and execution happens at machine speed without human review. An attacker who can place content into the pipeline's input stream by opening a pull request has a viable path to influence the agent's behavior.

Gemini CLI's design acknowledged this distinction by implementing a sandbox mode and, in interactive environments, prompting users to approve workspace trust before loading local configuration. The critical failure described in GHSA-wpqr-6v78-jr5g is that this safeguard was not applied in headless mode [2]. The trust decision—normally a human-in-the-loop checkpoint—was silently resolved in favor of whatever content occupied the workspace whenever the tool ran without a terminal attached. This vulnerability was discovered independently by Elad Meged of Novee Security and Dan Lisichkin of Pillar Security, and Google published its advisory and patches on April 29–30, 2026 [1][3][7].

The week of disclosure also saw reports of two separate vulnerabilities in the Cursor IDE: CVE-2026-26268 (CVSS 8.1), a git hook sandbox escape enabling arbitrary code execution through malicious `.git` configurations, and a separate credential exposure flaw (CVSS 8.2) allowing extensions to access API keys stored in SQLite [4]. The concurrent disclosure of similar vulnerability classes across multiple AI development tools suggests that pre-initialization code execution and configuration-layer trust boundaries represent a systemic gap in the current generation of AI coding agent design, across at least the tools examined in this disclosure cycle.

Security Analysis

The Workspace Trust Flaw

The primary vulnerability stems from how Gemini CLI handled workspace trust in non-interactive CI environments. Vulnerable versions of the tool automatically assumed that any workspace folder it ran inside was trusted for the purpose of loading configuration files and environment variables [1][2]. This meant that the contents of `.gemini/` directories—including settings files specifying which tools the agent may invoke, which environment variables to load, and which configuration overrides to apply—were read and applied without explicit approval.

In practice, an attacker who could place a malicious `.gemini/settings.json` or related configuration file into a repository by opening a pull request could cause that configuration to load the next time a workflow ran Gemini CLI against the repository's workspace. Because configuration loading occurs during agent initialization, malicious content executed before the sandbox ever came into effect. Novee Security researchers described the flaw as one in which attacker-controlled content was silently accepted as trusted configuration before any sandbox had initialized [1].

The CVSS 3.1 vector for this vulnerability reflects the severity of the design flaw: Attack Vector: Network, Attack Complexity: Low, Privileges Required: None, User Interaction: None, Scope: Changed, with High impact across Confidentiality, Integrity, and Availability, yielding a score of 10.0 [2]. The "None" ratings on Privileges Required and User Interaction capture the core problem—an external, unprivileged contributor with the ability to open a pull request can trigger exploitation without any further user interaction beyond the automated CI run.

CWE classifications for the advisory include CWE-20 (Improper Input Validation), CWE-77 (Command Injection), CWE-78 (OS Command Injection), and CWE-200 (Information Exposure) [2][7]. These designations reinforce that the vulnerability is a traditional software security failure—improper validation of untrusted input—applied to the novel context of AI agent configuration loading.

The --yolo Mode Allowlist Bypass

A secondary vulnerability compounded the workspace trust issue. Gemini CLI's `--yolo` flag—intended to enable fully automated, no-prompt execution—failed to enforce the tool allowlist defined in `settings.json` [2][3]. Administrators could configure fine-grained restrictions specifying which tools the agent was permitted to invoke, but these restrictions were silently ignored when `--yolo` was active. Generic entries such as `run_shell_command` in an allowlist acted as wildcards, permitting unrestricted shell execution regardless of more specific restrictions the operator had configured [8].

This meant that even organizations that had attempted to harden their Gemini CLI deployments through allowlisting were potentially exposed. The allowlist is the primary mechanism by which operators constrain agent capabilities in automated environments; bypassing it reduces the agent's security posture to that of an unrestricted shell executor carrying the full permissions of the CI runner. The patched version 0.39.1 corrects this behavior by evaluating tool allowlist restrictions regardless of the `--yolo` flag state [3].

Attack Chain in CI/CD Pipelines

The complete attack chain begins at the repository's external contribution surface and ends at the CI runner's execution environment [8]. An attacker submits a pull request containing a malicious `.gemini/` configuration directory. A workflow configured to run Gemini CLI against incoming pull requests—a common pattern for automated code review—triggers on the new PR. The CI runner checks out the attacker's branch, which includes the malicious configuration, and executes `gemini-cli` or the `run-gemini-cli` Action in the workspace. Because the tool is running headlessly, it automatically trusts the workspace and loads the attacker's configuration before any sandbox restriction applies. Arbitrary commands execute on the runner with the workflow's full permissions.

The following table summarizes the attack chain components and the specific control failures at each stage:

Stage	Attacker Input	Control Failure
Repository	Pull request with malicious `.gemini/` config	No PR content scanning for agent configuration files
CI Trigger	Workflow runs on PR head branch	Untrusted branch content reaches agent workspace
Agent Initialization	Config loaded pre-sandbox	Implicit workspace trust in headless mode (CWE-20)
Tool Execution	`--yolo` mode active	Allowlist not enforced; shell wildcards permitted (CWE-77, CWE-78)
Credential Access	Secrets in workflow environment	No isolation between untrusted processing and privileged execution

Once code executes on the runner, the attacker gains access to the full scope of the workflow's permissions: repository secrets, OAuth tokens, deploy keys, cloud credentials provisioned for the CI environment, and any downstream systems reachable from that execution context [5][6]. Researchers noted that the resulting access is sufficient for token theft, supply chain pivots into downstream deployments, and lateral movement to systems that trust the pipeline's identity [5][6].

Pre-Sandbox Execution: Why Traditional Defenses Fail

The defining characteristic of this vulnerability—and the reason it warrants particular attention from the security community—is that exploitation occurs before the agent's sandbox initializes. Sandbox isolation is a core runtime defense for AI coding agents, constraining which files the agent can read, which network connections it can make, and which commands it can execute. When configuration loading—a step occurring during agent startup—can be influenced by untrusted input, the sandbox never gets the opportunity to constrain the resulting behavior.

This represents a vulnerability class distinct from prompt injection. In a prompt injection attack, an attacker attempts to override the model's instructions through natural language embedded in data the model processes. In the Gemini CLI vulnerability, the exploitation path does not pass through the model at all: attacker-controlled content is loaded as trusted system configuration at the infrastructure layer, triggering command execution through the agent's own tooling framework before the model is ever invoked [1][8]. Researchers at Penligent described it as a source-to-sink problem in which "untrusted repository content plus agentic tool use plus insufficient execution boundaries" [8] combine to produce real command execution.

This distinction carries important implications for defensive posture. Prompt injection mitigations—system prompt hardening, output filtering, and model-level instruction following—provide no protection against this attack class. Effective defense requires treating the agent's configuration surface with the same rigor as the model's input surface: any file or environment variable that influences the agent's behavior must be considered a potential attack vector if it can be reached by untrusted external content. Human prompt instructions along the lines of "do not reveal secrets" provide no security boundary; actual protection requires token permissions, sandboxing, and architectural separation [8].

It is worth noting that a separate, earlier Gemini CLI vulnerability discovered by Tracebit and fixed in July 2025 (version 0.1.14) followed a different but related exploitation pattern: prompt injection through hidden instructions in a README file, combined with a command whitelist bypass that allowed chained shell execution [9]. That earlier flaw required the model to process and act on injected instructions; the GHSA-wpqr-6v78-jr5g vulnerability bypasses the model entirely. The progression from model-layer exploitation to infrastructure-layer exploitation across successive generations of the same tool illustrates how the AI coding agent attack surface continues to evolve.

Recommendations

Immediate Actions

Organizations running Gemini CLI in any CI/CD pipeline should treat this as a critical, time-sensitive remediation. The patched versions `@google/gemini-cli` 0.39.1 (or 0.40.0-preview.3) and `google-github-actions/run-gemini-cli` 0.1.22 address both the workspace trust bypass and the `--yolo` mode allowlist issue [2][3]. The GitHub Action updates automatically for workflows that reference it without a pinned version; organizations using pinned versions must manually update their workflow files. After upgrading, operators must make an explicit architectural decision about each CI workflow that invokes Gemini CLI: workflows processing trusted inputs only may set `GEMINI_TRUST_WORKSPACE: 'true'`, while workflows that process any untrusted input—including PRs from external contributors, forks, or public issue comments—must be configured without workspace trust and hardened per the guidance published in the advisory.

Short-Term Mitigations

Beyond patching, organizations should audit their Gemini CLI deployments against a set of hardening principles that this vulnerability made concrete. Any workflow that processes untrusted repository content should operate with read-only GitHub tokens; secrets required for deployment or downstream access should be held in a separate, trusted workflow stage that runs only after untrusted-content analysis is complete [8]. The `--yolo` flag should be removed from any workflow that may encounter externally contributed content; if fully automated execution is required, tool allowlists must specify narrow, explicit tool names rather than generic `run_shell_command` entries that act as unrestricted wildcards.

Organizations should also review whether their workflow configurations scan for the presence of `.gemini/` directories in incoming pull requests. A PR that introduces agent configuration files warrants elevated review scrutiny, particularly in repositories where CI runs with elevated permissions. Enforcing ephemeral, isolated runners for workflows that process public input limits the blast radius if exploitation occurs: a compromised ephemeral runner cannot serve as a persistent foothold, and its credential scope should be minimal by design. Container-based isolation between the untrusted-content processing stage and any stage requiring privileged credentials provides an additional layer of defense.

Strategic Considerations

This vulnerability points to a broader architectural pattern that security teams should address before deploying additional AI coding agents in sensitive environments. The root problem is not specific to Gemini CLI; it is a consequence of embedding agents that load configuration from the workspace into pipelines that also process untrusted content from that workspace. Any AI agent that reads configuration from the repository it is analyzing—a pattern found in multiple AI coding tools—creates a potential path from external contributor to agent configuration to code execution.

Security architects should establish a clear separation principle for agentic CI/CD workflows: the workspace used for untrusted content analysis should never contain configuration that influences the agent's trust model or tool permissions. Where possible, agent configuration should be loaded exclusively from locations outside the repository being analyzed—runner-level environment variables, secrets management systems, or a separate trusted configuration repository. This architectural separation substantially reduces exposure to this class of vulnerability, rather than relying on correct trust handling by the agent at initialization time.

Organizations should also evaluate the credential scope granted to CI workflows that run AI agents. The principle of least privilege applies with particular force to agentic workflows: an agent reviewing external pull requests requires no deployment credentials, database access, or push rights to production branches. Scoping workflow tokens to the minimum permissions required for the analysis task limits the value of any credential exfiltration that does occur. As AI coding agents become more capable and more deeply integrated into software delivery pipelines, the attack surface they represent grows correspondingly—and the cost of a credential compromise at that layer scales with the downstream systems those credentials can reach.

CSA Resource Alignment

This vulnerability directly engages several layers of the MAESTRO framework for agentic AI threat modeling. Layer 6 (Deployment Infrastructure) addresses the risks introduced when AI agents operate within CI/CD pipelines that carry privileged execution context; the workspace trust bypass exploited exactly the trust assumptions that MAESTRO identifies as requiring explicit threat modeling in automated deployment environments [10]. Layer 7 (Agent Ecosystem) addresses the risks of agents interacting with external, untrusted content as part of their normal operation—precisely the scenario that made pull request-based exploitation possible [10]. Security teams applying MAESTRO to their Gemini CLI deployments should evaluate the automatic workspace trust behavior as a control gap under headless execution conditions—a risk that falls directly within Layer 6's scope.

CSA's February 2026 analysis applying MAESTRO to real-world CI/CD agent threat models addressed the CI/CD agent threat surface from a MAESTRO perspective, describing CI/CD pipelines as a high-priority threat surface for agentic AI because the combination of untrusted input processing, privileged execution context, and automated operation creates conditions where a single configuration trust failure can cascade into full pipeline compromise [12]. The Gemini CLI vulnerability is consistent with that analysis and provides a concrete, production-observed instance of the threat model.

The CSA AI Controls Matrix (AICM) provides a complementary set of controls relevant to this incident [11]. The AICM's supply chain security domain addresses controls for verifying the integrity of inputs to AI systems, including configuration files and agent initialization parameters. The tool execution and access management domains address the principle of least privilege for agent tool calls—the control gap that the `--yolo` allowlist bypass exploited. Organizations conducting AICM-based assessments of their AI agent deployments should specifically evaluate controls in the Supply Chain, Access Management, and Agent Execution domains as directly implicated by GHSA-wpqr-6v78-jr5g.

More broadly, this incident is consistent with CSA's guidance that the agent's trust boundary must be defined architecturally, not at the model level. The Gemini CLI case demonstrates that a successful attack does not require passing through the model at all: infrastructure-level configuration loading, if reachable by untrusted content, is sufficient for code execution. This finding should inform how security teams evaluate any AI coding agent that reads configuration from the repository it analyzes—a design pattern that creates an equivalent exposure path wherever the workspace contains untrusted external content.

References

[1] Elad Meged, Novee Security. "A CVSS 10.0 in Gemini CLI: How Agentic Workflows Are Reshaping Supply Chain Risk." Novee Security Blog, April 2026.

[2] GitHub. "Gemini CLI: Remote Code Execution via workspace trust and tool allowlisting bypasses · GHSA-wpqr-6v78-jr5g." GitHub Advisory Database, April 2026.

[3] Google / google-github-actions. "Update to Gemini CLI and run-gemini-cli Trust Model." GitHub Security Advisory GHSA-wpqr-6v78-jr5g, April 2026.

[4] The Hacker News. "Google Fixes CVSS 10 Gemini CLI CI RCE and Cursor Flaws Enable Code Execution." The Hacker News, April 2026.

[5] The Register. "Google's fix for critical Gemini CLI bug might break your CI/CD pipelines." The Register, April 30, 2026.

[6] SecurityWeek. "Critical Gemini CLI Flaw Enabled Host Code Execution, Supply Chain Attacks." SecurityWeek, April 2026.

[7] CSO Online. "Max-severity RCE flaw found in Google Gemini CLI." CSO Online, April 2026.

[8] Penligent. "Gemini CLI RCE, Workspace Trust and the CI/CD Agent Attack Surface." Penligent, April 2026.

[9] Tracebit. "Code Execution Through Deception: Gemini AI CLI Hijack." Tracebit Blog, June 2025.

[10] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[11] Cloud Security Alliance. "AI Controls Matrix." CSA Artifacts, July 2025.

[12] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threat Models: From Framework to CI/CD Pipeline." CSA Blog, February 2026.

