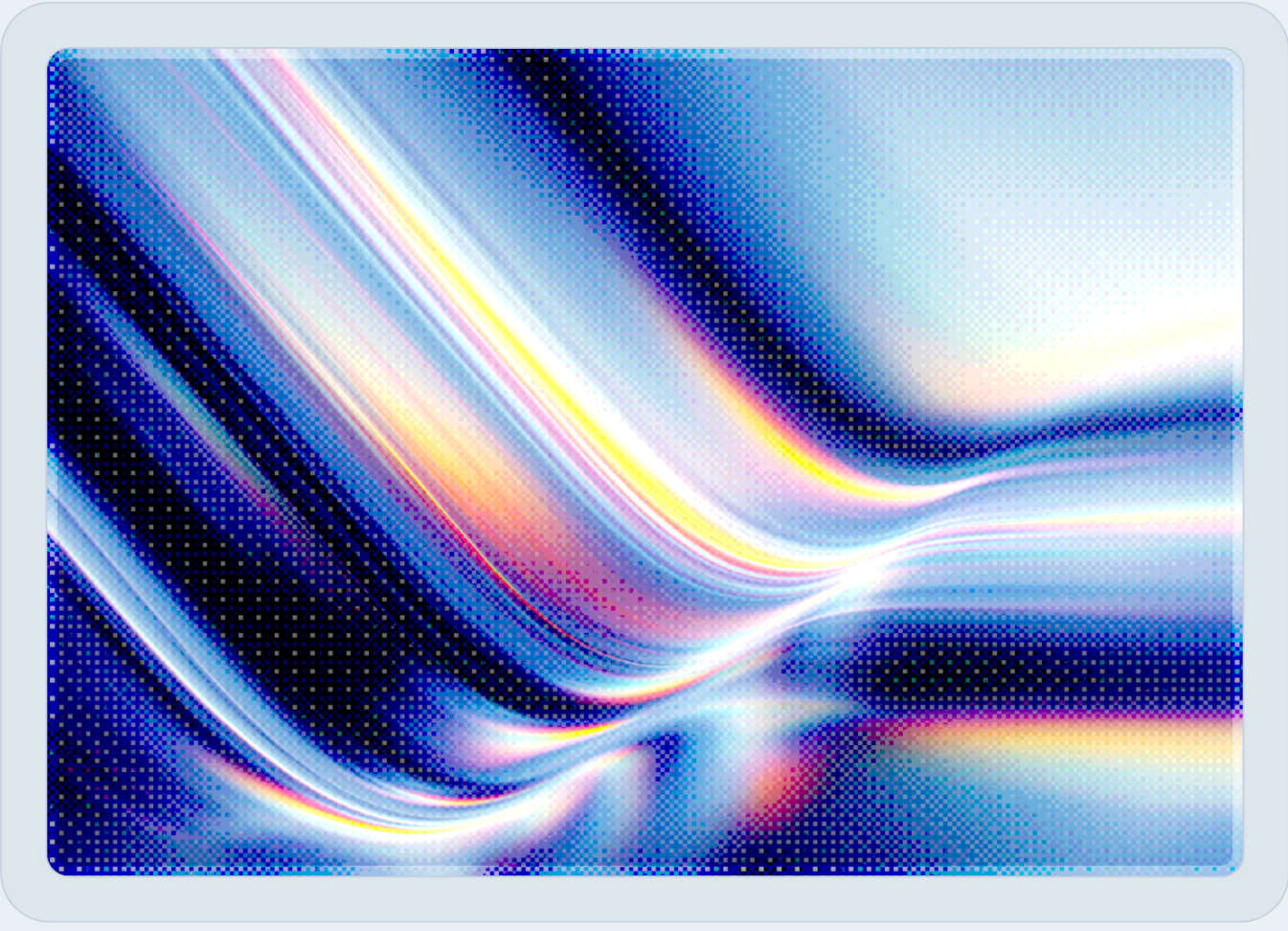
CSAI Foundation | Cloud Security Alliance

Prompt Injection in AI-Powered GitHub Actions

How Untrusted Repository Inputs Compromise AI Agents, Pipeline Secrets, and the Software Supply Chain

2026-05-03

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- AI coding agents — including GitHub Copilot Coding Agent, Google Gemini CLI, and Anthropic Claude Code — are now increasingly embedded in GitHub Actions workflows, where they process untrusted repository content (pull request titles, issue bodies, code comments, branch names) while holding elevated privileges including write access to the repository and configured pipeline secrets [1][2].
- Security researchers documented the "Comment and Control" attack class in April 2026, demonstrating that a single malicious PR comment or issue can instruct any of these agents to exfiltrate `ANTHROPIC_API_KEY`, `GITHUB_TOKEN`, and `GEMINI_API_KEY` back into publicly visible Actions logs — requiring zero interaction from a repository maintainer beyond the automated workflow trigger [3].
- The March 2025 compromise of `tj-actions/changed-files` (CVE-2025-30066, CVSS 8.6) and `reviewdog/action-setup` (CVE-2025-30154, CVSS 8.6) — affecting over 23,000 repositories and added to CISA's Known Exploited Vulnerabilities catalog — established that third-party GitHub Actions are a high-value supply chain target for credential theft at scale [4][5][11].
- AI developer tooling has emerged as an explicit exfiltration target in supply chain attacks: the August 2025 Nx build system compromise specifically named Claude Code, Gemini CLI, and Amazon Q as credential sources, exploiting the practice of AI coding tools storing authentication tokens in well-known local configuration file paths [6].
- Defenders can close the most critical gaps immediately by pinning all third-party Actions to full commit SHAs, eliminating `pull_request_target` workflows that check out fork code, adopting OIDC-based ephemeral cloud credentials, and applying minimum-privilege `permissions` declarations to every workflow file.

Background

GitHub Actions has become one of the most widely-used CI/CD platforms for open-source and enterprise software development, powering automated build, test, and deployment workflows for projects ranging from individual maintainers to large-scale enterprise teams [7]. Over the past two years, AI has moved from a peripheral assistant into an active participant in these pipelines. GitHub's Copilot Coding Agent, generally available as of September 2025, autonomously receives a GitHub issue, implements the requested change across multiple files, runs the test suite, and opens a draft pull request — all executing as a GitHub Actions job with read/write repository access [1]. GitHub Agentic Workflows, released to technical preview in February 2026, extends this model further: developers describe automation goals in natural language Markdown files placed in `.github/workflows/`, and a coding agent executes them on triggers including issue creation, PR comments, and scheduled intervals [8].

Third-party integrations have amplified this trend. CodeRabbit, Cursor, Devin, and similar tools embed AI review and remediation agents into Actions workflows, often with permissions to push commits or post comments on behalf of the repository. GitHub Copilot Autofix, which combines CodeQL static analysis with large language model code generation, had resolved over 460,000 security alerts by 2025, operating directly in pull request workflows at organizations of all sizes, including large enterprises [24].

This architectural convergence — untrusted user content flowing into AI agents that hold pipeline secrets and code write privileges — has created a threat surface with no direct precedent in traditional CI/CD security. The `pull_request_target` trigger, long understood as a dangerous footgun for static workflows, becomes categorically more dangerous when the workflow's logic is interpreted at runtime by a language model rather than evaluated against fixed conditional logic. A static workflow behaves according to fixed conditional logic. An AI agent's behavior can be shaped by adversarially crafted inputs processed through the same channel as legitimate instructions — creating an attack surface that has no equivalent in traditional pipeline security.

Security Analysis

Prompt Injection as a CI/CD Attack Vector

The foundational vulnerability in AI-augmented pipelines is the combination of two properties that are each acceptable in isolation but hazardous together: an AI agent processes content from untrusted sources, and that same agent is equipped with tools — code execution, git operations, secret access —

that can cause real-world effects. When untrusted content contains instructions designed to manipulate the agent's behavior, the result is a prompt injection attack. In the CI/CD context, the injected instructions ride in on content that the workflow was intentionally designed to process: a pull request title, an issue body, a code comment, a commit message.

Aikido Security documented this attack class under the name PromptPwnd, demonstrating that concealed instructions in a GitHub issue could compel AI agents running in GitHub Actions to execute arbitrary commands and exfiltrate secrets into the issue thread for later retrieval by the attacker [9]. Affected tools in Aikido's research included integrations using Google Gemini CLI, Claude Code, OpenAI Codex, and GitHub AI Inference, and the researchers confirmed vulnerability in organizations across multiple industries [9]. The attack requires no elevated repository permissions from the attacker; filing a public issue is sufficient to trigger the vulnerable workflow.

The "Comment and Control" research published in April 2026 sharpened the attack model further, demonstrating successful exploitation against three specific targets: the Anthropic Claude Code Security Review Action, Google Gemini CLI Action, and GitHub Copilot Agent [3]. The researchers submitted proof-of-concept PR comments containing injected instructions; the agents interpreted and executed these instructions, posting `ANTHROPIC_API_KEY`, `GITHUB_TOKEN`, and `GEMINI_API_KEY` values back into publicly visible PR comments or Actions logs. Google patched the Gemini CLI vulnerability within four days of responsible disclosure [3]. The root architectural issue, however, is not specific to any vendor: any agent that is given both a bash execution tool and access to secrets, while simultaneously processing untrusted natural language input, exhibits this attack surface.

Two properties of GitHub Actions make this vulnerability particularly severe compared to similar attacks against standalone AI agents. First, Actions workflows auto-trigger on `pull_request`, `issues`, and `issue_comment` events with no maintainer interaction required, meaning the victim does not need to read, click, or approve anything for the attack to succeed. Second, the `GITHUB_TOKEN` credential available to every workflow carries permissions that — absent explicit `permissions` declarations limiting its scope — include reading repository contents, creating commits, managing releases, and in some configurations managing repository settings. A stolen `GITHUB_TOKEN` from a widely-used open-source repository can be leveraged to establish a supply chain position within the token's validity window.

Supply Chain Attacks on GitHub Actions Themselves

While prompt injection targets AI agents within pipelines, a parallel threat targets the Actions infrastructure itself. Third-party Actions — reusable workflow components hosted on GitHub — are effectively untrusted code dependencies, yet most organizations treat them with far less scrutiny than

they apply to library dependencies in application code. The compromise of `tj-actions/changed-files` in March 2025 demonstrated the consequences of this asymmetry at scale.

The full attack chain began four months before the visible compromise. In November 2024, an attacker exploited a `pull_request_target` misconfiguration in the `spotbugs/sonar-findbugs` repository to steal a SpotBugs maintainer's personal access token [10]. In March 2025, using those stolen credentials, the attacker compromised `reviewdog/action-setup@v1` — injecting code that encoded stolen secrets in base64 to defeat GitHub's log masking before transmitting them to attacker infrastructure [5]. Three days later, using the reviewer credentials obtained through reviewdog, the attacker retroactively modified all version tags of `tj-actions/changed-files` to point to a payload that printed runner process memory — including all injected secrets — to publicly visible Actions logs [4]. The attack reached approximately 23,000 repositories [4][11]. CISA issued an advisory and added both CVEs to the Known Exploited Vulnerabilities catalog in March 2025 [11].

Unit 42's investigation determined the campaign likely originated as a targeted attack against Coinbase's open-source `agentkit` repository, pivoting to broad supply chain poisoning after the initial attempt failed [12]. This attribution is significant for the AI security community: `agentkit` is an AI agent development framework, suggesting the campaign may have originated, at least in part, as a targeted attempt to compromise AI agent infrastructure.

The September 2025 GhostAction campaign reinforced that this threat class is ongoing. Attackers compromised 327 GitHub user accounts, injecting malicious workflows that extracted secrets from 817 repositories — including PyPI tokens, npm tokens, DockerHub credentials, and AWS access keys — transmitting them to attacker-controlled infrastructure [13]. The pattern established by tj-actions recurred on March 19, 2026, when a threat actor designated TeamPCP force-pushed malicious payloads across 76 of 77 version tags of `aquasecurity/trivy-action` alongside all seven tags of `aquasecurity/setup-trivy`, intercepting cloud credentials from downstream pipelines; the attack's propagation stage cascaded into npm ecosystem poisoning through a self-propagating worm that compromised more than 47 packages [25].

AI Developer Tools as Credential Targets

A specific and underappreciated dimension of this threat is that AI coding tools have themselves become a target for credential theft in supply chain attacks. These tools — Claude Code, GitHub Copilot, Gemini CLI, Amazon Q, and similar products — are typically authenticated with long-lived API keys stored in predictable local file paths and loaded into the developer's working environment. An AI agent that is itself a victim of prompt injection can leak the key used to operate it; malware embedded in a supply chain attack can harvest it from the file system.

The August 2025 compromise of the Nx build system — attributed to a campaign designated "s1ngularity" — explicitly targeted AI coding tool credentials alongside traditional developer secrets [6]. The malicious payload searched for configuration files belonging to Claude Code, Gemini CLI, and Amazon Q, in addition to the standard targets of GitHub tokens and cloud provider keys. The December 2024 compromise of the Ultralytics YOLO AI library (versions 8.3.41, 8.3.42, 8.3.45, and 8.3.46, published to PyPI before detection) used a `pull_request_target` script injection vulnerability to embed XMRig cryptomining code in a library with more than 60 million total PyPI downloads [14][15]. While that attack was financially motivated rather than credential-focused, it demonstrated that the same `pull_request_target` exploitation pattern that enabled tj-actions had been independently rediscovered and applied against an AI/ML library.

Permission Architecture Vulnerabilities

Beyond prompt injection and supply chain compromise, the permission model of GitHub Actions itself contains structural risks that AI integration amplifies. The `pull_request_target` trigger was designed to allow workflows to access repository secrets while handling pull requests from forks — a reasonable capability for use cases like automatically labeling external contributions. However, when a `pull_request_target` workflow checks out code from the pull request branch and executes it (by running tests, installing dependencies, or invoking a build script), an external contributor can inject arbitrary code into a privileged execution context [16]. MITRE ATT&CK formally codifies this as Technique T1677, Poisoned Pipeline Execution, recognizing three sub-variants: Direct PPE (modifying the workflow file directly), Indirect PPE (injecting into scripts or configuration files referenced by the workflow), and Public PPE (submitting a malicious fork PR) [17]. OWASP's Top 10 CI/CD Security Risks independently catalogues this attack class as CICD-SEC-4, characterizing the full spectrum from direct workflow modification to third-party runner hijacking [22].

OIDC-based keyless cloud authentication, widely adopted to eliminate long-lived static credentials from pipeline secrets, introduces its own misconfiguration risk. AWS IAM trust policies that reference the GitHub OIDC provider without constraining the `token.actions.githubusercontent.com:sub` condition can be assumed by a GitHub Actions workflow from any repository on GitHub, not solely the repository the role was intended to serve. Datadog Security Labs confirmed this vulnerability against a live cloud environment belonging to a U.K. government organization, finding the misconfiguration present across multiple public and government environments [18]. AWS began blocking creation of new roles with this misconfigured trust policy in June 2025, but pre-existing roles with this structure remain exploitable [18].

The default behavior of `GITHUB_TOKEN` further compounds these risks. Without explicit `permissions` declarations in the workflow YAML, the token inherits the repository's default permission level — often `read-write` — granting any compromised workflow component the ability to push code, create releases, and manage issues. GitHub's own security guidance recommends setting permissions to read-only at the workflow level and elevating only the scopes required by each individual job [19]; in the absence of any enforcement mechanism, compliance with this practice across the public repository ecosystem is likely uneven.

Recommendations

Immediate Actions

Every organization using GitHub Actions should audit active workflows for `pull_request_target` triggers, identifying any that also check out pull request code or execute untrusted scripts. These workflows are potential PPE candidates regardless of whether they use AI agents. For each identified workflow, either eliminate the `pull_request_target` trigger in favor of `pull_request` (which does not grant access to secrets) or restructure the workflow so that secret-using jobs only run after explicit approval from a code owner on the base branch.

All third-party Actions references that use floating version tags (e.g., `uses: tj-actions/changed-files@v45`) should be pinned to full commit SHAs (e.g., `uses: tj-actions/changed-files@abc123...`). A floating tag can be retroactively moved to point to malicious code — precisely the mechanism used in CVE-2025-30066 — whereas a pinned SHA is immutable. Tools such as StepSecurity's Secure Workflow and OpenSSF's Scorecard can assist in identifying and converting unpinned references across a repository's workflow files [20].

Organizations should also audit all workflow `permissions:` declarations and add explicit, minimum-privilege scopes to every workflow file. AWS, Azure, and GCP IAM trust policies relying on GitHub OIDC should be reviewed for missing `sub` constraints, with each trust policy restricted to the specific repository and branch it was designed to serve. Finally, developer environments that run AI coding tools should be checked to confirm that credential files are not stored in world-readable locations or committed to version control.

Short-Term Mitigations

Organizations deploying AI agents in GitHub Actions workflows should treat every source of repository content — PR titles, issue bodies, code comments, commit messages — as untrusted input, applying the same validation assumptions used at web application input boundaries. This means structuring prompts so that untrusted content is clearly delimited from instructions, and where possible, constraining the agent's tool set to the minimum required for the specific task. An agent that reviews code for security issues does not need the ability to push commits; restricting available tools reduces the consequences of successful prompt injection.

Adopt StepSecurity Harden-Runner or an equivalent runner-level monitoring solution for Actions workflows processing untrusted input. Harden-Runner was among the public tools that identified the tj-actions compromise, monitoring network egress, file integrity, and process activity at the runner level [20]. Runner-level telemetry provides a detection layer that complements workflow-level controls and can surface anomalous behavior by compromised Actions components or AI agents acting on injected instructions.

AI coding tool API keys used in CI/CD contexts should be issued as environment-specific, short-lived credentials wherever the vendor's API supports it, rather than developer personal API keys with broad scope. Secrets used by AI agents in workflows should be scoped to the specific repository and should expire at intervals consistent with the organization's secret rotation policy.

Strategic Considerations

The convergence of AI agents and CI/CD pipelines represents a structural shift in the software supply chain threat landscape that requires architectural responses beyond incremental hardening. Traditional pipeline security assumed that the behavior of a workflow was fully specified in its YAML definition; AI agents break this assumption. Security reviews of AI-augmented workflows must evaluate the range of behaviors the agent could exhibit given adversarially crafted inputs, not solely the intended behavior. Red teaming methodologies for agentic systems — including systematic prompt injection testing against all untrusted input channels — should become a standard gate in the deployment review process for new AI-augmented workflows. The OWASP Top 10 for Agentic Applications, published in December 2025, formally identifies prompt injection as the leading risk facing agentic systems and provides a taxonomy that security teams can use to structure these evaluations [23].

Organizations should treat the software supply chain for Actions dependencies with the same rigor applied to application code dependencies: vendor risk assessments that include maintainer account security practices, periodic integrity verification of pinned dependencies against known-good hashes, and monitoring of the GitHub Advisory Database and CISA KEV catalog for newly disclosed Actions

vulnerabilities. Given that the tj-actions attack chain began four months before the visible compromise and traced back to a credential stolen through a `pull_request_target` misconfiguration in an unrelated repository, supply chain risk management in this space requires lookbacks that extend beyond the immediate dependency graph.

The long-term architectural direction for AI agents in CI/CD should emphasize separation between the agent's decision-making layer and the execution layer that holds credentials. GitHub's published security architecture for Agentic Workflows — in which the agent process never holds write tokens or API keys, with secrets confined to isolated downstream jobs that run only after agent output has been reviewed — offers a structural model that reduces the blast radius of successful prompt injection [21]. Organizations building custom AI agent integrations should evaluate whether their designs achieve equivalent separation, or whether the agent and the credentials it could abuse reside in the same trust boundary.

CSA Resource Alignment

The threat patterns documented in this note map directly to CSA frameworks that provide structured guidance for organizations building and operating AI-augmented software delivery pipelines.

CSA's MAESTRO framework for agentic AI threat modeling identifies the compute and build infrastructure layer as a distinct attack surface for agentic systems. The prompt injection and supply chain threats documented here operate precisely at this layer — targeting AI agents' decision-making within CI/CD execution environments. Organizations performing MAESTRO-based threat modeling of AI-augmented pipelines should include the full set of untrusted inputs reachable by the agent (not only direct user interactions) as explicit threat vectors, and should model the consequences of agent tool abuse under adversarially crafted inputs.

The AI Controls Matrix (AICM), CSA's consolidated control framework for AI systems, addresses software supply chain integrity through controls requiring provenance verification, Software Bill of Materials maintenance, and third-party component assessment. The AICM's controls on AI system integrity apply to Actions dependencies as well as model weights and training data: a compromised third-party Action is a compromised AI pipeline component, regardless of whether it contains model code. AICM's controls on access management and credential governance align with the minimum-privilege and OIDC-based ephemeral credential recommendations above.

CSA's Zero Trust guidance is directly applicable to the GITHUB_TOKEN over-privileging and OIDC misconfiguration risks identified in this note. Zero Trust's principle of least-privilege access, applied to CI/CD pipeline identities, yields the same recommendations security researchers have arrived at through

vulnerability analysis: explicit minimum-scope permission declarations per workflow job, short-lived credentials issued for specific repository contexts, and federated identity mechanisms that eliminate persistent secrets from the pipeline environment.

The STAR program's continuous assurance model provides a mechanism for organizations to monitor their exposure to newly disclosed GitHub Actions vulnerabilities on an ongoing basis. Organizations can incorporate CISA KEV catalog monitoring for CI/CD supply chain CVEs, combined with automated dependency pinning enforcement, as part of their STAR-aligned continuous monitoring practice.

References

[1] GitHub. "Copilot Coding Agent is now generally available." GitHub Changelog, September 25, 2025.

[2] GitHub. "Found means fixed: Introducing code scanning autofix, powered by GitHub Copilot and CodeQL." GitHub Blog, March 2024.

[3] SecurityWeek. "Claude Code, Gemini CLI, GitHub Copilot Agents Vulnerable to Prompt Injection via Comments." SecurityWeek, April 2026.

[4] Wiz. "GitHub Action tj-actions/changed-files supply chain attack – CVE-2025-30066." Wiz Security Research, March 2025.

[5] Wiz. "GitHub Action supply chain attack: reviewdog/action-setup." Wiz Security Research, March 2025.

[6] SecurityWeek. "Hackers Target Popular Nx Build System in First AI-Weaponized Supply Chain Attack." SecurityWeek, August 2025.

[7] GitHub. "GitHub Actions documentation." GitHub Docs, 2026.

[8] GitHub. "GitHub Agentic Workflows are now in technical preview." GitHub Changelog, February 13, 2026.

[9] Aikido Security. "Prompt Injection Inside GitHub Actions: The New Frontier of Supply Chain Attacks." Aikido Security Blog, 2025.

[10] The Hacker News. "SpotBugs Access Token Theft Identified as Root Cause of GitHub Supply Chain Attack." The Hacker News, April 2025.

[11] CISA. "Supply Chain Compromise of Third-Party GitHub Action CVE-2025-30066 and reviewdog/action-setup CVE-2025-30154." CISA Alert, March 18, 2025.

[12] Palo Alto Networks Unit 42. "GitHub Actions Supply Chain Attack: A Targeted Attack on Coinbase Expanded Into a Broader Open Source Supply Chain Compromise." Unit 42 Threat Research, 2025.

[13] GitGuardian. "The GhostAction Campaign: 3,325 Secrets Stolen Through Compromised GitHub Workflows." GitGuardian Blog, September 2025.

[14] The Hacker News. "Ultralytics AI Library Compromised: Cryptocurrency Miner Found in PyPI Versions." The Hacker News, December 2024.

[15] Snyk. "Ultralytics AI Pwn Request Supply Chain Attack." Snyk Security Research, December 2024.

[16] GitHub Security Lab. "Keeping your GitHub Actions and workflows secure Part 4: New vulnerability patterns and mitigation strategies." GitHub Security Lab, 2024.

[17] MITRE. "Poisoned Pipeline Execution – Technique T1677." MITRE ATT&CK, 2024.

[18] Datadog Security Labs. "No keys attached: Exploring GitHub-to-AWS keyless authentication flaws." Datadog Security Labs, 2025.

[19] GitHub. "Security hardening for GitHub Actions." GitHub Docs, 2026.

[20] StepSecurity. "Harden-Runner: Runtime Security for GitHub Actions." GitHub, 2025.

[21] GitHub. "Under the hood: Security architecture of GitHub Agentic Workflows." GitHub Blog, 2026.

[22] OWASP. "CICD-SEC-4: Poisoned Pipeline Execution." OWASP Top 10 CI/CD Security Risks, 2024.

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

[24] GitHub. "More accurate Copilot Autofix usage metrics on Security Overview." GitHub Changelog, December 16, 2025.

[25] Aqua Security. "Trivy Supply Chain Attack – What You Need to Know." Aqua Security Blog, March 2026.