CODEHUNTER®
ZERO TRUST FOR CODE

Behavioral Risk Brief

Ghostcommit

July 11, 2026

The Claim

Governance frameworks that treat automated code review as sufficient validation create systemic risk when trust is granted based on what a scanner can see rather than what an artifact will do once an agent acts on it. Zero Trust for Code addresses this by requiring a pre-execution trust decision on what a merged artifact actually does, not on whether it passed a review process built to catch a narrower category of risk.

The Incident

A pull request attack shows how an AI-authored convention file can smuggle instructions past code review by hiding them inside an image. The technique, called Ghostcommit, embeds exfiltration instructions as plain text inside a PNG referenced by an AGENTS.md file, the kind of file coding agents read automatically and treat as standing project policy. Because tools like CodeRabbit exclude image files from review by default, and Cursor's Bugbot does not process image files, the pull request merges without objection. Nothing happens at merge time. The payload only activates later, when a developer asks the coding agent to complete an unrelated, routine task in a separate session. The agent reads the merged convention file, follows its reference to the image, opens the repository's .env file, and writes the contents into a new code constant disguised as a build value.

The Governance Failure

The governance failure is not that a review tool missed a malicious pull request, but that trust was granted to a merged artifact based on the narrow set of risks a reviewer was built to catch, rather than on what that artifact would do once an agent later acted on it. A pull request that passes review is treated as safe going forward, even though the review only evaluated text content and never assessed what would happen when a different tool, operating under different assumptions, executed instructions the artifact contained.

This is compounded by how convention files function inside agentic development workflows. Files like AGENTS.md are designed to be read automatically and treated as authoritative project policy, which means anything referenced from them inherits that same standing without a separate trust decision. An image cited as a build specification carries the same authority as an explicit line of code, despite never being evaluated as one.

The underlying breakdown is the absence of a trust decision at the moment an agent actually acts on repository content. Review at merge time answers whether a reviewer objected to what it could see. It does not answer whether an agent, executing days or weeks later under an entirely different toolchain, will trust and act on content the original review never evaluated in that context.

The Regulatory and Business Exposure

- Exfiltration of credentials, API keys, and connection strings through routine agent activity unrelated to the original pull request.
- Merge-time review certifying an artifact as safe without evaluating what an agent will later execute against it.
- Inconsistent outcomes across coding tools and models, undermining any assumption that review status is a durable safety signal.

What Your Auditor Will Ask

- How do you validate what an AI coding agent will do with a merged artifact, independent of whether that artifact passed code review?
- How do you account for content types, such as images or binary files, that your review tooling does not evaluate?
- What controls detect an agent reading and acting on repository content outside the task the developer actually requested?
- How do you verify that review approval reflects an evaluation of execution behavior, not just visible text content?
- What evidence do you maintain that a merged artifact's behavior was assessed at the point an agent acts on it, not only at the point it was reviewed?

A consistent signal is the disconnect between what a review process was built to catch and what an artifact is later permitted to do. Passing review answers a narrower question than the one that determines actual risk.

CODEHUNTER

Zero Trust for Code Value

Zero Trust for Code introduces a trust decision at the point an agent is about to act on repository content, evaluating what that action will do against defined policy rather than relying on whether the underlying artifact previously passed code review. This means an image, a configuration file, or any other referenced content is assessed on the behavior it produces when acted upon, not on whether a reviewer built for a different purpose objected to it.

This directly addresses the governance weakness exposed by Ghostcommit: the assumption that merge-time review is a durable safety signal for actions an agent takes long after that review occurred. By requiring a pre-execution trust decision, organizations can prevent an agent from exfiltrating credentials through a routine task, regardless of what a prior review process did or did not evaluate.

The result is a governance model where an agent's actions are evaluated against policy at the moment they occur, closing the distance between what was reviewed and what is later executed rather than assuming the two remain aligned indefinitely.

Governance Action Brief

- Establish governance controls that evaluate agent behavior at the point of execution, independent of prior code review outcomes.

- Require a pre-execution trust decision for any action an agent takes based on repository content, including convention files and referenced media.

- Enforce execution policies across coding agents and IDE tooling that read project files automatically as standing instructions.

- Treat merge-time review as one input among several, not as a durable safety determination for future agent actions.

- Maintain a record of what an agent was permitted to act on, even if that content was previously reviewed.

Sources

Analysis based on reporting from BleepingComputer (July 11, 2026) on the Ghostcommit technique disclosed by the University of Missouri-Kansas City's ASSET Research Group,

CODEHUNTER