INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-11 06:46 UTC

Ghostcommit: Image-Embedded Prompt Injection Bypasses AI Code Review to Exfiltrate Secrets

SECURITY ANALYSIS | **HIGH** | CVSS 7.5

SCC Item ID	SCC-STY-2026-0346
Type	Security Analysis
Severity	HIGH
CVSS Base Score	7.5
Affected Products	CodeRabbit, Bugbot, AI coding agents (general); repositories containing .env secret files in CI/CD pipelines
Published	2026-07-11T05:03:57
Discovery Source	Rss

Executive Summary

Researchers published a proof-of-concept attack, dubbed 'Ghostcommit,' demonstrating that prompt injection instructions hidden inside PNG image files can direct AI code review agents, specifically CodeRabbit and Bugbot, to locate and exfiltrate secrets from .env files, returning stolen credentials as obfuscated numeric data to evade detection. The technique exploits a structural blind spot: AI coding agents trust submitted pull request artifacts and apply no input sanitization to non-code file types, meaning the attack surface is the review pipeline itself. Combined with prior disclosures of RCE and write-access exploitation paths against CodeRabbit, this signals a maturing pattern of security research against AI-assisted development tooling, a category now deeply embedded in enterprise CI/CD workflows with privileged repository access.

Technical Analysis

The Ghostcommit technique works by embedding natural-language prompt injection directives inside PNG image files attached to pull requests. When an AI coding agent, CodeRabbit and Bugbot were the confirmed affected tools, processes the PR, it reads the image content and interprets the embedded text as legitimate instructions. According to BleepingComputer's reporting on the research, the hidden instructions direct the agent to locate .env files within the repository and read their contents. Exfiltrated credentials are then returned as obfuscated numeric data, a deliberate evasion technique that maps to MITRE ATT&CK T1027 (Obfuscated Files or Information). The attack exploits two compounding weaknesses: first, AI code review agents extend implicit trust to all artifacts in a pull request, not just code files; second, neither CodeRabbit nor Bugbot applied input sanitization or content inspection to non-code file types such as images, per the research findings. This absence of sanitization is the structural root cause. The attack chain aligns with T1552.001 (Unsecured Credentials: Credentials In Files) as the collection objective and T1195.001 (Supply Chain Compromise: Development Tools) as the broader category. The AI agent functions as an unwitting execution proxy for the injected instructions, analogous to command execution but mediated through the agent's legitimate processing logic. No CVE has been assigned, reflecting the novelty of the vulnerability class; prompt injection in agentic pipelines does not map cleanly to traditional software flaw taxonomy. The Ghostcommit disclosure does not stand alone. Kudelski Security previously documented RCE and write-access exploitation paths against CodeRabbit, and Endor Labs published a 'PwnedRabbit' analysis examining GitHub App permission abuse by the same tool. Together, these disclosures describe a pattern: AI coding agents operate with broad repository permissions, process diverse and partially untrusted input types, and have historically lacked the input validation and least-privilege design that security teams expect from tools with this level of access. The industry implication extends beyond CodeRabbit and Bugbot. Any AI agent that processes pull request artifacts, images, documents, configuration files, and acts on their content without sanitization is structurally susceptible to this class of attack. As agentic AI becomes standard in development pipelines, the attack surface for prompt injection expands proportionally.

Action Checklist

1. Step 1: Assess exposure, determine whether your organization uses CodeRabbit, Bugbot, or any other AI coding agent that processes pull request artifacts (including images and non-code files) within your CI/CD pipelines.
2. Step 2: Review controls, apply least-privilege principles to AI agent repository permissions (NIST AC-6); restrict agent access to .env files and other credential stores; exclude image and non-code file types from AI agent processing scope by default; if inclusion is required, enforce strict input sanitization and content inspection before ingestion; enforce secret storage in dedicated vaults rather than .env files in-repo (aligned with CWE-522 mitigation).
3. Step 3: Update threat model, incorporate image-embedded prompt injection as an attack vector in your CI/CD and supply chain threat model; map to MITRE ATT&CK T1195.001 (Supply Chain Compromise: Development Tools), T1552.001 (Credentials In Files), and T1027 (Obfuscated Files or Information); document AI coding agents as privileged pipeline actors requiring their own threat surface assessment.
4. Step 4: Communicate findings, brief engineering leadership and the AppSec team on the specific risk: AI agents with broad repository read access processing untrusted file types can be weaponized to exfiltrate credentials; frame this as a pipeline integrity issue, not solely a vendor bug, since no CVE exists and vendor patches may be delayed.
5. Step 5: Monitor developments, track CodeRabbit and Bugbot changelogs and security advisories for input sanitization or image content inspection updates; monitor Endor Labs, Kudelski Security, and BleepingComputer for follow-on research in this tooling category; watch for CISA or NIST guidance on agentic AI security as this vulnerability class matures.

IR / Forensic Enrichment

Triage Priority	URGENT

Escalation Criteria	Escalate immediately to CISO and legal/compliance if forensic review of historical AI agent PR review comments reveals numeric-encoded or obfuscated strings consistent with credential exfiltration, if any `.env` secrets confirmed accessible to CodeRabbit or Bugbot are also used in production systems, or if the organization operates in a regulated sector (PCI-DSS, HIPAA, SOC 2) where CI/CD credential exposure triggers breach notification obligations.
Recovery Notes	After restricting AI agent access and rotating all secrets that were stored in `.env` files accessible to CodeRabbit or Bugbot, verify that no active sessions or API tokens derived from those secrets remain live by auditing downstream service authentication logs for the affected credentials. Monitor AI agent PR review output for at least 30 days post-containment for anomalous structured data in review comments — specifically numeric sequences, base64 fragments, or unusual formatting that was not present before the Ghostcommit PoC was published. Update your CI/CD pipeline acceptance criteria to require human AppSec review of any PR that introduces non-code file types (images, archives, binaries) until vendor-confirmed input sanitization controls are in place for your deployed AI agents.
Forensic Artifacts	GitHub PR review comment exports from CodeRabbit and Bugbot for all PRs submitted in the 90 days prior to detection — specifically searched for numeric sequences, comma-delimited digit strings, or base64-like blocks in review output that match the Ghostcommit obfuscated exfiltration encoding pattern described in the Endor Labs PoC | GitHub Audit Log exports (Organization Settings → Audit Log) filtered for CodeRabbit and Bugbot app activity, capturing all repository read events, file access events, and OAuth token usage timestamps to establish whether `.env` files were accessed during AI agent review sessions | Repository file tree snapshots showing the presence, path, and last-modified timestamps of `.env`, `.env.local`, `.env.production`, and similar credential files at HEAD and in PR branch history — captured via `git log --all --full-history -- '**/.env*'` to reconstruct what was accessible to the AI agent at the time of each PR review | CI/CD pipeline YAML files (`.github/workflows/*.yml`, `.gitlab-ci.yml`, `Jenkinsfile`) and AI agent configuration files (`.coderabbit.yaml`, Bugbot integration config) preserved as-found — these establish the agent's configured scope of file access and processing permissions at the time of the potential compromise | PNG and other non-code files submitted in pull requests during the exposure window — extracted via `git show :` and analyzed with `strings`, `exiftool`, and a text dump of embedded metadata to identify hidden prompt injection payloads matching the Ghostcommit technique of embedding natural-language instructions in image file content or metadata

Per-Action IR Details

Step 1: Assess exposure — determine whether your organization uses CodeRabbit, Bugbot, or any other AI coding agent that processes pull request artifacts (including images and non-code files) within your CI/CD pipelines.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establishing IR capability through asset awareness and pipeline inventory

Controls: CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run `git log --all --name-only | grep -E '\.(png|jpg|gif|webp|bmp)'` across active repos to identify image file submissions in PR history. Cross-reference with CI/CD pipeline YAML files (`grep -r 'coderabbit\|bugbot\|ai-review' .github/workflows/`) to enumerate AI agent integrations. A two-person team can complete this audit in 2-4 hours per organization.

Evidence: This step does not alter live state. Capture the current list of active CI/CD workflow files (`.github/workflows/*.yml`, `.gitlab-ci.yml`, `Jenkinsfile`), AI agent configuration files (`.coderabbit.yaml`, any Bugbot config), and installed GitHub App or OAuth app permissions from the repository settings before any changes are made. Document the scope of repository access each AI agent holds.

Step 2: Review controls — apply least-privilege principles to AI agent repository permissions (NIST AC-6); restrict agent access to .env files and other credential stores; verify that image and non-code file types are excluded from AI agent processing scope or sanitized before ingestion; enforce secret storage in dedicated vaults rather than .env files in-repo (aligned with CWE-522 mitigation).

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: Restricting attacker-controllable access paths to limit further credential exposure

Controls: NIST AC-6 (Least Privilege), NIST AC-3 (Access Enforcement), CIS 3.3 (Configure Data Access Control Lists), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts)

Compensating: Add a `.coderabbitignore` or equivalent exclusion config to block AI agent processing of `*.png`, `*.jpg`, `*.gif`, `*.env`, `*.env.*`, and `secrets/` paths. Add a pre-commit hook using `git-secrets` or `trufflehog` (`trufflehog git file://. --only-verified`) to detect secrets committed to `.env` files before they reach the repo. Rotate any `.env` secrets to a secrets manager (HashiCorp Vault free tier, AWS Secrets Manager, or GitHub Encrypted Secrets) and add `.env` to `.gitignore` globally via `git config --global core.excludesfile ~/.gitignore_global`.

Evidence: BEFORE revoking or restricting AI agent permissions, capture the current OAuth token scopes and GitHub App installation permissions for CodeRabbit and Bugbot from the repository's Installed GitHub Apps settings page. Export the current `.coderabbit.yaml` configuration and any Bugbot workflow triggers. Capture a snapshot of which `.env` files exist in the repository at HEAD (`find . -name '.env*' -not -path './.git/*'`) and whether they contain live credentials (`trufflehog filesystem . --json > pre-containment-secrets-scan.json`). These establish a baseline for what was accessible to the AI agent before restriction.

Step 3: Update threat model — incorporate image-embedded prompt injection as an attack vector in your CI/CD and supply chain threat model; map to MITRE ATT&CK T1195.001 (Supply Chain Compromise: Development Tools), T1552.001 (Credentials In Files), and T1027 (Obfuscated Files or Information); document AI coding agents as privileged pipeline actors requiring their own threat surface assessment.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Updating organizational threat models and detection capabilities based on observed attack techniques

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Create a dedicated threat model entry using STRIDE or a simple attack tree in a Markdown file committed to your security runbook repo. Document the Ghostcommit attack chain: malicious PNG submitted via PR → AI agent ingests image → injected prompt instructs agent to read `.env` → agent returns secrets as obfuscated numeric encoding in PR comment or review output. Tag CodeRabbit and Bugbot as 'privileged pipeline actors' in your asset inventory (CIS 1.1) and assign them a dedicated threat surface review on the same cadence as third-party code dependencies.

Evidence: This step does not alter live state. Before finalizing the updated threat model, collect and preserve any existing PR review comments from CodeRabbit or Bugbot on PRs that included PNG or non-code file submissions. Export GitHub PR review history via the GitHub REST API (`GET /repos/{owner}/{repo}/pulls/{pull_number}/reviews`) for the lookback period. Review for anomalous AI agent output patterns: numeric sequences, base64-like strings, or unusually structured review comments that could represent obfuscated credential exfiltration already in progress.

Step 4: Communicate findings — brief engineering leadership and the AppSec team on the specific risk: AI agents with broad repository read access processing untrusted file types can be weaponized to exfiltrate credentials; frame this as a pipeline integrity issue, not solely a vendor bug, since no CVE exists and vendor patches may be delayed.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Communicating incident scope and impact to authorized stakeholders to enable coordinated response

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting)

Compensating: Prepare a one-page impact brief using the Ghostcommit PoC disclosure from Endor Labs as the primary reference. Include: (1) which AI agents are deployed in your pipelines, (2) which repositories have `.env` files accessible to those agents, (3) whether any PRs containing PNG files were reviewed by the agent in the past 90 days. Distribute via your existing incident communication channel (email, Slack `#security-alerts`). No SIEM required — this is evidence-gathering and stakeholder notification, achievable with GitHub API queries and manual log review.

Evidence: This step does not alter live state. Before the briefing, pull and preserve the AI agent review logs for all PRs in the past 90 days that included image file attachments. Use the GitHub API: `GET /repos/{owner}/{repo}/pulls?state=all` filtered for PRs with image-type file changes, then `GET /repos/{owner}/{repo}/pulls/{pull_number}/files` to identify PNG/image submissions. Export all associated CodeRabbit or Bugbot review comments for forensic review — specifically looking for numeric sequences, hex-encoded strings, or structured data blocks in review output that could represent exfiltrated `.env` values encoded as described in the Ghostcommit PoC.

Step 5: Monitor developments — track CodeRabbit and Bugbot changelogs and security advisories for input sanitization or image content inspection updates; monitor Endor Labs, Kudelski Security, and BleepingComputer for follow-on research in this tooling category; watch for CISA or NIST guidance on agentic AI security as this vulnerability class matures.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Using lessons learned to improve detection, update policies, and integrate threat intelligence for emerging attack classes

Controls: CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Set up RSS or GitHub watch notifications on the CodeRabbit (`coderabbitai/coderabbit`) and Bugbot release pages. Create a free CISA Known Exploited Vulnerabilities (KEV) feed subscription at cisa.gov/known-exploited-vulnerabilities-catalog. Establish a monthly review task to check Endor Labs blog and NVD for any CVE assignment related to prompt injection in AI code review agents. Write a YARA rule targeting PR comment output logs for numeric-sequence exfiltration patterns (e.g., sequences of 3-digit numbers separated by commas or spaces that map to ASCII/decimal-encoded credential characters) and run it against archived PR review comment exports.

Evidence: This step does not alter live state. Establish a log retention baseline now: configure GitHub audit log export (available under Organization Settings → Audit Log → Export) to capture all AI agent activity going forward, and retain for a minimum of 90 days per your logging policy. Archive the current Ghostcommit PoC technical writeup from Endor Labs and Kudelski Security as reference artifacts in your threat intelligence repository. These preserved exports become the detection baseline against which future AI agent behavior anomalies will be compared.

Detection Guidance

Log and alert on AI code review agent activity that involves file reads outside the diff scope of a pull request, specifically any agent action that reads files not present in the changed file set. Monitor for .env file access events in CI/CD pipeline logs that are not attributable to explicit pipeline steps. Review AI agent output for numeric-encoded or obfuscated strings in review comments, particularly patterns inconsistent with normal code review language; this maps to the T1027 obfuscation technique described in the research. Audit pull request artifacts for image files in repositories where images are not a normal part of the project type, an unexpected PNG in a backend service PR is anomalous. Apply NIST AU-2 (Event Logging) and AU-6 (Audit Record Review, Analysis, and Reporting) to CI/CD pipeline activity, ensuring AI agent actions are logged with sufficient fidelity to reconstruct what files were accessed and what output was produced. Use CIS 8.2 (Collect Audit Logs) as a baseline to confirm pipeline-layer logging is enabled. Behavioral hunt hypothesis: query pipeline logs for AI agent comment events that contain long runs of digits or base-encoded strings, which may indicate numeric obfuscation of exfiltrated credential material. The cited BleepingComputer source and linked research may contain specific indicators, consult those sources directly for any payload signatures or example obfuscation patterns published by the researchers.

Framework Mappings

MITRE-ATTACK

- **T1027** — Obfuscated Files or Information
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059** — Command and Scripting Interpreter
- **T1552.001** — Credentials In Files
- **T1190** — Exploit Public-Facing Application
- **T1566** — Phishing

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-8** — Spam Protection
- **IA-5** — Authenticator Management
- **SI-10** — Information Input Validation
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A03:2021** — Injection
- **A01:2021** — Broken Access Control

CIS-V8

- **5.2** — Use Unique Passwords
- **16.10** — Apply Secure Design Principles in Application Architectures

- **6.3** — Require MFA for Externally-Exposed Applications
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(a)(1)** — Access Control
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1027	Obfuscated Files or Information	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1552.001	Credentials In Files	Credential-Access
T1190	Exploit Public-Facing Application	Initial-Access
T1566	Phishing	Initial-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/ghostcommit-hides-pr...	T2
CodeRabbit changelog: new features, updates, and releases	https://docs.coderabbit.ai/changelog	T3
When CodeRabbit became PwnedRabbit - Endor Labs	https://www.endorlabs.com/learn/when-coderabbit-became-pwnedrabbit-...	T3
CodeRabbit Security Posture: How we safeguard your repositories	https://coderabbit.ai/blog/our-security-posture-how-we-safeguard-yo...	T3

Source	URL	Tier
How We Exploited CodeRabbit: From a Simple PR to RCE and Write ...	https://kudelskisecurity.com/research/how-we-exploited-coderabbit-f...	T3

Generated 2026-07-11 06:46 UTC by TJS Security Command Center