INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-28 13:44 UTC

CVE-2026-3854: Git Push Injection Exposes GitHub's Server-Side Pipeline to Unauthenticated RCE

CVE VULNERABILITY | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CVE-2026-0086
Type	CVE Vulnerability
CVE ID	CVE-2026-3854
Severity	CRITICAL
CVSS Base Score	9.5
EPSS Score	0.0039 (60th percentile)
Affected Products	GitHub.com (patched), GitHub Enterprise Cloud and variants (patched), GitHub Enterprise Server prior to versions 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, 3.20.0
Published	2026-04-28T15:30:00+00:00
Discovery Source	Rss:T1 Psirt

Executive Summary

A critical remote code execution vulnerability in GitHub's git push pipeline allowed any authenticated user with push access to execute arbitrary commands on GitHub's backend servers. GitHub.com and Enterprise Cloud variants were patched within two hours of the March 4, 2026 disclosure; however, GitHub Enterprise Server customers running versions prior to 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, or 3.20.0 remain exposed until local patches are applied. Organizations using self-hosted GitHub Enterprise Server face potential compromise of their entire software development pipeline, including source code, CI/CD secrets, and downstream supply chain integrity.

Technical Analysis

CVE-2026-3854 is a CVSS 9.5 critical OS command injection vulnerability (CWE-78, CWE-20) in GitHub's server-side git push processing pipeline. The flaw stems from insufficient input validation and improper neutralization of special characters passed via git push options, allowing unsanitized input to reach OS-level command execution on backend servers. Any authenticated user holding push access to any repository is a valid attacker, no elevated privilege required beyond standard contributor access. GitHub.com and all Enterprise Cloud variants received emergency patches within two hours of responsible disclosure on March 4, 2026.

GitHub Enterprise Server customers must manually apply patches across six supported release trains: 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, and 3.20.0. MITRE technique coverage includes T1190 (Exploit Public-Facing Application), T1059 (Command and Scripting Interpreter), T1072 (Software Deployment Tools), and T1195.002 (Compromise Software Supply Chain), T1552.001 (Credentials in Files). No exploitation has been confirmed by GitHub at time of disclosure. EPSS score is 0.391% (60th percentile), indicating moderate near-term exploitation probability relative to the broader CVE population. Sources: GitHub Security Blog; NVD CVE-2026-3854.

Action Checklist

1. **Containment:** Immediately identify all GitHub Enterprise Server instances in your environment and determine their current version. Isolate any instance running a version prior to 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, or 3.20.0 from external network access or restrict push access to trusted users only until patching is complete. GitHub.com and Enterprise Cloud customers require no action; patches were applied by GitHub on March 4, 2026.
2. **Detection:** Review GitHub Enterprise Server audit logs for anomalous push events, unexpected process spawns from the git push pipeline, or unusual OS-level activity on the server host. Look for push operations containing special characters (semicolons, backticks, pipe symbols, dollar signs) in push option fields. If your Enterprise Server runs on a host with EDR coverage, hunt for child processes spawned from git-related parent processes (e.g., git-receive-pack) that are not part of normal hook execution. Correlate with T1059 and T1190 behavioral patterns.
3. **Eradication:** Apply the appropriate GitHub Enterprise Server patch for your release train: 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, or 3.20.0. Follow GitHub's official upgrade documentation for your version. Do not skip release trains without consulting GitHub's upgrade path guidance. After patching, rotate any secrets, tokens, or credentials that were stored on or accessible from the GitHub Enterprise Server host, as server-side RCE would have granted access to the host filesystem and environment variables.
4. **Recovery:** After patch application, confirm the installed version matches the target patched release via the GitHub Enterprise Server admin console or API. Re-enable push access for previously restricted users and repositories. Monitor push pipeline activity for 72 hours post-patch for any residual anomalous behavior. Validate that no unauthorized repository webhooks, deploy keys, OAuth tokens, or GitHub Actions runners were added during the exposure window.
5. **Post-Incident:** Audit your GitHub Enterprise Server network exposure: determine whether the instance was internet-facing or restricted to internal networks, and whether push access was granted broadly or scoped narrowly. Review your software supply chain risk posture; this vulnerability could have enabled silent code injection into repositories, with downstream impact on build artifacts. Evaluate whether T1195.002 (software supply chain compromise) controls are adequate, including code signing, artifact integrity verification, and CI/CD pipeline audit logging. Document time-to-patch as a metric for future SLA benchmarking.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO and legal/compliance immediately if forensic evidence indicates successful exploitation of CVE-2026-3854 (child processes from git-receive-pack, modified hook scripts, or unauthorized credentials) — a confirmed or suspected RCE on a GHES host that stores source code, secrets, or CI/CD pipeline configurations constitutes a potential data breach and software supply chain compromise event that may trigger breach notification obligations under applicable regulations (GDPR Article 33, state breach laws) and requires supply chain impact assessment per NIST SA-12 (Supply Chain Protection).
Recovery Notes	Post-patch, monitor GHES audit logs and host-level process logs for a minimum of 72 hours, specifically watching for git-receive-pack child process anomalies and any new webhook or deploy key registrations that were not authorized through your normal access-granting process. If exploitation during the exposure window cannot be ruled out, treat all secrets, tokens, and credentials accessible from the GHES host filesystem and environment as compromised and complete rotation before re-enabling broad push access. Validate the integrity of build artifacts produced from repositories hosted on the affected GHES instance during the exposure window before promoting them to production or distributing them downstream.
Forensic Artifacts	GitHub Enterprise Server audit log API export (GET /api/v3/enterprise/audit-log?phrase=action:git.push) filtered from 2026-03-04T00:00:00Z to patch timestamp — specifically entries where push_options fields contain shell metacharacters (semicolons, backticks, pipe symbols, dollar signs) indicative of CVE-2026-3854 push option injection | Linux auditd execve records (type=EXECVE in /var/log/audit/audit.log) where ppid matches a git-receive-pack process — unexpected child processes (e.g., /bin/sh, curl, wget, python3) spawned from git-receive-pack are the primary indicator of successful RCE via this vulnerability's push pipeline injection mechanism | GHES pre-receive hook scripts at /data/user/git-hooks/ and /etc/github/pre-receive-hooks/ — file modification timestamps (stat output) and content hashes post-exploitation would reflect attacker-inserted persistence; diff against any configuration backup taken before 2026-03-04 | Process environment dump from /proc/[git-receive-pack PID]/environ captured during or after the exposure window — CVE-2026-3854 server-side RCE would have granted read access to environment variables including GITHUB_TOKEN, AWS credentials, database connection strings, and other secrets injected into the GHES process environment at runtime | Repository webhook and deploy key creation audit records from all GHES organizations and repositories (GET /api/v3/repos/:owner/:repo/hooks and /keys) with created_at timestamps falling within the 2026-03-04 to patch-apply window — unauthorized additions represent post-exploitation persistence mechanisms consistent with MITRE T1546 (Event-Triggered Execution) via git hooks or T1098 (Account Manipulation) via deploy key injection

Per-Action IR Details

Containment — Immediately identify all GitHub Enterprise Server instances in your environment and determine their current version. Isolate any instance running a version prior to 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, or 3.20.0 from external network access or restrict push access to trusted users only until patching is complete. GitHub.com and Enterprise Cloud customers require no action — patches were applied by GitHub on March 4, 2026.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), NIST CM-2 (Baseline Configuration), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Use the GitHub Enterprise Server management console (https://HOSTNAME:8443) or run 'curl -s http://HOSTNAME/setup/api/settings | python3 -m json.tool | grep version' to enumerate installed versions across all instances. For network isolation without an enterprise firewall, apply host-based iptables rules to block inbound TCP 22 (git over SSH) and TCP 443 (git over HTTPS) from untrusted CIDR ranges: 'iptables -I INPUT -p tcp --dport 443 -s 0.0.0.0/0 -j DROP && iptables -I INPUT -p tcp --dport 22 -s 0.0.0.0/0 -j DROP' — then explicitly allow only trusted admin subnets. If full isolation is not feasible, use the GitHub Enterprise Server site admin console to set the instance to maintenance mode, which blocks all git push operations while preserving read access.

Evidence: Before isolating, capture a point-in-time snapshot of the GitHub Enterprise Server audit log via the API ('GET /api/v3/enterprise/audit-log?phrase=action:git.push&per_page=100') and export to a tamper-evident location. Record the current GHES version from the admin console, the list of all repositories with external push access enabled, and all configured pre-receive hooks from the site admin panel at /setup/hooks — pre-receive hooks are the execution context most likely exploited by CVE-2026-3854's push injection mechanism.

Detection — Review GitHub Enterprise Server audit logs for anomalous push events, unexpected process spawns from the git push pipeline, or unusual OS-level activity on the server host. Look for push operations containing special characters (semicolons, backticks, pipe symbols, dollar signs) in push option fields. If your Enterprise Server runs on a host with EDR coverage, hunt for child processes spawned from git-related parent processes (e.g., git-receive-pack) that are not part of normal hook execution. Correlate with T1059 and T1190 behavioral patterns.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: If no EDR is present on the GHES host, deploy Sysmon on Windows-based GHES hosts using SwiftOnSecurity's config, specifically enabling ProcessCreate (Event ID 1) with ParentImage filters matching '*git-receive-pack*' or '*git-upload-pack*'. On Linux-based GHES hosts (most common), enable auditd with the rule '-a always,exit -F arch=b64 -S execve -F ppid=$(pgrep -f git-receive-pack) -k ghes_push_rce' to capture child process execution from the git push daemon. Query the GHES audit log API with: 'curl -H "Authorization: token ADMIN_PAT" "https://HOSTNAME/api/v3/enterprise/audit-log?phrase=action:git.push+pull_request:false" | jq ".[] | select(.data.push_options != null)"' and pipe through grep for shell metacharacters: 'grep -P "[;`|$\(\)]"+'. Use the free Sigma rule ss7b4c1d (process injection via git hooks) converted to auditd format for continuous monitoring.

Evidence: Collect the following before proceeding to eradication: (1) Full GHES audit log export covering the disclosure window from 2026-03-04T00:00:00Z forward via 'GET /api/v3/enterprise/audit-log?phrase=action:git.push&after=2026-03-04T00:00:00Z' — preserve in WORM storage or a signed export. (2) On the GHES Linux host, capture '/var/log/auth.log' and '/var/log/syslog' for the same window, plus 'ps auxf' output showing the current git-receive-pack process tree. (3) Dump the contents of '/data/user/git-hooks/' and '/etc/github/pre-receive-hooks/' to identify any hook scripts added or modified after 2026-03-04. (4) Capture environment variables visible to the git push pipeline process via 'sudo cat /proc/$(pgrep git-receive-pack)/environ | tr "\0" "\n"' — a successful RCE exploit of CVE-2026-3854 would have exposed these variables, including any injected secrets.

Eradication — Apply the appropriate GitHub Enterprise Server patch for your release train: 3.14.25, 3.15.20, 3.16.16, 3.17.13, 3.18.8, 3.19.4, or 3.20.0. Follow GitHub's official upgrade documentation for your version. Do not skip release trains without consulting GitHub's upgrade path guidance. After patching, rotate any secrets, tokens, or credentials that were stored on or accessible from the GitHub Enterprise Server host, as server-side RCE would have granted access to the host filesystem and environment variables.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Download the GHES hotpatch package directly from GitHub's release page (https://enterprise.github.com/releases) — verify the SHA256 checksum of the '.pkg' or '.ova' file before applying: 'sha256sum ghes-3.X.Y.pkg'. Apply via the admin console upgrade path at https://HOSTNAME:8443/setup/packages/new or via ghe-upgrade CLI: 'ghe-upgrade /path/to/ghes-3.X.Y.pkg'. For credential rotation without a secrets manager, enumerate all environment variables from '/etc/github/' config files and '.env' files under '/data/user/' using 'sudo grep -rE "(TOKEN|SECRET|PASSWORD|KEY)" /etc/github/ /data/user/common/ --include="*.conf" --include="*.env"' — document each and rotate through the originating system (GitHub App credentials via the app settings UI, deploy keys via repository settings, PATs via user account settings).

Evidence: Before applying the patch, take a full filesystem snapshot or VM snapshot of the GHES host to preserve forensic state — this is critical if exploitation is suspected. Capture 'find /data/user/git-hooks/ /tmp/ /var/tmp/ -newer /var/log/github/gitrpcd.log -type f -ls' to identify files written during the exploitation window. Export the complete list of OAuth tokens, GitHub Actions runner registration tokens, and deploy keys active during the exposure window via the GHES API ('GET /api/v3/enterprise/settings/hooks', 'GET /api/v3/repos/:owner/:repo/keys') — these represent the blast radius of a successful RCE that read environment variables from the git push pipeline process.

Recovery — After patch application, confirm the installed version matches the target patched release via the GitHub Enterprise Server admin console or API. Re-enable push access for previously restricted users and repositories. Monitor push pipeline activity for 72 hours post-patch for any residual anomalous behavior. Validate that no unauthorized repository webhooks, deploy keys, OAuth tokens, or GitHub Actions runners were added during the exposure window.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-3 (Configuration Change Control), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Confirm patched version via: 'curl -s -H "Authorization: token ADMIN_PAT" https://HOSTNAME/api/v3/meta | jq .installed_version'. To audit for backdoored webhooks added during the exposure window, run: 'curl -H "Authorization: token ADMIN_PAT" "https://HOSTNAME/api/v3/repos/:owner/:repo/hooks" | jq ".[] | {id, name, config, created_at}"' and filter for creation timestamps after 2026-03-04T00:00:00Z. For Actions runner audit, query: 'GET /api/v3/enterprises/:enterprise/actions/runners' and cross-reference against your known-good runner inventory baseline. Use a simple bash script iterating over all repos via the GHES API to enumerate deploy keys and webhooks created in the exposure window — a 2-person team can run this in under an hour across most GHES instances.

Evidence: Capture a post-patch version confirmation screenshot or API response and append to incident record. Export the full list of webhooks, deploy keys, OAuth applications, and GitHub Actions runners with creation timestamps from the GHES API for all repositories and organizations — diff this against the pre-incident baseline captured during containment. Review '/var/log/github/gitrpcd.log' for the 72-hour post-patch window, specifically filtering for any process spawn anomalies that would indicate a persistent hook or cron job installed by a successful pre-patch exploit.

Post-Incident — Audit your GitHub Enterprise Server network exposure: determine whether the instance was internet-facing or restricted to internal networks, and whether push access was granted broadly or scoped narrowly. Review your software supply chain risk posture — this vulnerability could have enabled silent code injection into repositories, with downstream impact on build artifacts. Evaluate whether T1195.002 (software supply chain compromise) controls are adequate, including code signing, artifact integrity verification, and CI/CD pipeline audit logging. Document time-to-patch as a metric for future SLA benchmarking.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST RA-3 (Risk Assessment), NIST SA-12 (Supply Chain Protection), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-11 (Audit Record Retention), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: To assess supply chain impact, generate a git log diff for all repositories active during the exposure window: 'git log --all --since="2026-03-04" --until="PATCH_DATE" --format="%H %an %ae %s" > exposure_window_commits.txt' and review for commits by unknown authors or with suspicious messages. For artifact integrity, if Sigstore or similar is not in use, compute and compare SHA256 hashes of build artifacts produced during the exposure window against known-good baselines: 'sha256sum /path/to/artifact > artifact.sha256'. Document GHES network exposure by reviewing your firewall or cloud security group rules — capture whether TCP 443 and TCP 22 were reachable from 0.0.0.0/0 at time of disclosure. Record time-to-patch (disclosure 2026-03-04 to patch apply timestamp) as a measured SLA data point in your vulnerability management tracking.

Evidence: Retain all audit log exports, process snapshots, network flow logs, and git log diffs from the exposure window for a minimum of 12 months per NIST AU-11 (Audit Record Retention) to support any future supply chain compromise investigations. Archive the pre- and post-patch GHES configuration exports from the admin console. Preserve a copy of the GHES version manifest file at time of discovery ('/data/user/common/github.conf' or equivalent) as evidence of the unpatched state for regulatory or insurance purposes.

Detection Guidance

On GitHub Enterprise Server hosts, examine audit logs for push events containing non-standard characters in push option payloads. Focus on the git-receive-pack process and any processes it spawned. If host-level logging is available (syslog, auditd, or EDR telemetry), search for unexpected child process creation from git-related parent processes, particularly shell invocations (bash, sh, cmd) not associated with configured server-side hooks. Review environment variable access and file reads from the git push pipeline process context, which could indicate credential harvesting consistent with T1552.001. No public IOCs or exploitation indicators have been released as of March 4, 2026 disclosure. No CISA KEV listing at time of this report. EPSS at 60th percentile suggests active exploitation is plausible in the near term; treat absence of confirmed exploitation as temporary, not permanent.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`No confirmed exploitation IOCs available at time of disclosure`	GitHub stated no exploitation was confirmed as of March 4, 2026 responsible disclosure. No IP addresses, hashes, or domain indicators have been published. Monitor GitHub Security Blog and NVD for updates.	LOW

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1650** — Acquire Access
- **T1195.002** — Compromise Software Supply Chain
- **T1552.001** — Credentials In Files
- **T1072** — Software Deployment Tools

- **T1059** — Command and Scripting Interpreter

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A03:2021** — Injection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **2.5** — Allowlist Authorized Software
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.26** — Application security requirements
- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1650	Acquire Access	Resource-Development
T1195.002	Compromise Software Supply Chain	Initial-Access
T1552.001	Credentials In Files	Credential-Access
T1072	Software Deployment Tools	Execution
T1059	Command and Scripting Interpreter	Execution

Sources

Source	URL	Tier
The latest security news for developers - The GitHub Blog	https://github.blog/security/securing-the-git-push-pipeline-respond...	T3
	https://github.blog/security/securing-the-git-push-pipeline-respond...	T3
	https://thehackernews.com/2026/01/aws-codebuild-misconfiguration-ex...	T3
	https://gbhackers.com/pcpcat-malware/	T3
CVE-2026-3854 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-3854	T1

Generated 2026-04-28 13:44 UTC by TJS Security Command Center