INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-03 13:58 UTC

Unpatched VS Code Zero-Day Exposes GitHub OAuth Tokens via Webview Abuse, PoC Live, No CVE Assigned

CVE VULNERABILITY | HIGH | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0260
Type	CVE Vulnerability
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Visual Studio Code (all versions, Microsoft); github.dev (GitHub/Microsoft)
Published	2026-06-03T02:50:30
Discovery Source	Rss

Executive Summary

A publicly disclosed, unpatched zero-day in Visual Studio Code allows an attacker to steal a developer's GitHub OAuth token with a single click on a malicious link. The stolen token grants full access to every private GitHub repository the victim can reach, inheriting the full permission scope of the victim's GitHub account. No patch is available from Microsoft, a working exploit is publicly circulating, and no CVE has been assigned, meaning automated vulnerability scanners will not flag this exposure.

Technical Analysis

The vulnerability abuses VS Code's sandboxed webview message-passing system. Specifically, the postMessage API between webviews and the VS Code extension host can be manipulated to silently install a rogue extension without user confirmation. That extension then exfiltrates the victim's GitHub OAuth token. The attack chain aligns with CWE-284 (improper access control), CWE-522 (insufficiently protected credentials), CWE-79 (cross-site scripting via webview), and CWE-345 (insufficient verification of data authenticity). MITRE ATT&CK coverage includes T1528 (Steal Application Access Token), T1176 (Browser Extensions, applicable to webview context), T1059.007 (JavaScript execution), T1204.001 (Malicious Link), T1550.001 (Application Access Token reuse), T1195.001 (Compromise Software Dependencies), and T1566.002 (Spearphishing Link). Affected scope: all current VS Code versions and the github.dev web-based editor. The stolen token is not scoped to a single repository, it inherits the full permission set of the victim's GitHub account. No CVE assigned. No patch released as of 2026-03-04. A public proof-of-concept is available (credited to researcher Ammar Askar; corroborating technical detail at aikido.dev). CVSS base estimated at 7.5 (High). EPSS and KEV data are

not yet populated, scanners will not surface this automatically. Note: Community reports of fake VS Code vulnerability posts have circulated; this advisory is grounded in public technical research and not yet confirmed by Microsoft.

Action Checklist

1. Containment, Immediately restrict VS Code's ability to open untrusted external links: enforce the 'security.workspace.trust' setting and set 'extensions.autoUpdate' to false in enterprise VS Code policy. Revoke and rotate any GitHub OAuth tokens issued to developer accounts that use VS Code or github.dev, prioritizing accounts with access to sensitive or production repositories. This aligns with NIST AC-17 (Remote Access restrictions) and credential rotation controls.
2. Detection, Review git provider audit logs for OAuth token usage from unexpected IP addresses, geographies, or user agents. In GitHub, navigate to Settings > Security log and filter for 'oauth_access' events. Review VS Code extension telemetry and installed extension lists on developer workstations for unrecognized extensions (CIS 2.1, Software Inventory; system file analysis controls). Search endpoint logs for VS Code webview network traffic to non-Microsoft, non-GitHub domains. No CVE or specific event ID is available yet, behavioral indicators are the primary detection path.
3. Eradication, No vendor patch is available. As a compensating control, enforce extension allowlisting via VS Code's 'extensions.allowedExtensionIDs' policy setting and restrict webview API access where enterprise policy permits. Remove any unrecognized VS Code extensions from developer machines (CIS 2.3, Address Unauthorized Software). Revoke all GitHub Personal Access Tokens and OAuth tokens for affected developers and reissue with minimum-necessary scopes per NIST AC-6 (Least Privilege). Apply user account permission controls to limit GitHub token scope organization-wide.
4. Recovery, After token rotation, verify repository access audit logs show no anomalous clone, download, or settings-change events in the 30 days prior to detection. Re-enable only allow-listed VS Code extensions. Confirm enterprise VS Code policy changes have propagated to all developer workstations (NIST CM, Configuration Management baseline). Monitor GitHub organization audit logs for minimum 30 days post-remediation for residual token reuse (NIST AU-6, Audit Record Review, Analysis, and Reporting).
5. Post-Incident, This vulnerability exposed gaps in developer workstation extension governance (CIS 2.1, CIS 2.3) and OAuth token scoping discipline (NIST AC-6, CIS 6.1). Initiate a review of all GitHub OAuth token grants across the organization, enforce least-privilege token scoping as a standing policy, and add VS Code extension inventory to the quarterly asset review cycle (CIS 1.1). Consider whether github.dev access should be restricted via network policy until a vendor patch is available. Track the Microsoft VS Code security advisory page and CVE assignment for this issue; patch immediately upon release.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to CISO and legal counsel if GitHub audit logs confirm any 'git.clone' or 'repo.download' event against a private repository from an unrecognized IP or user-agent during the exposure window, as this indicates confirmed data exfiltration of potentially proprietary source code, credentials embedded in code, or regulated data, triggering breach notification assessment under applicable privacy regulations.

Recovery Notes	After completing token rotation and policy enforcement, conduct a repository-by-repository review of any private repos accessible to affected developer accounts, checking GitHub's audit log for clone, download, and webhook-creation events in the 30 days preceding detection — attacker dwell time between token theft and discovery is unknown given the absence of a CVE or vendor alert. Monitor GitHub organization audit logs daily for at least 30 days post-remediation specifically for OAuth token reuse from unexpected sources, as attackers may have cached stolen tokens prior to revocation or harvested tokens from multiple developers across an extended campaign. Do not restore github.dev access or re-enable VS Code extension auto-update until Microsoft publishes a confirmed patch and your team has validated it addresses the webview token exfiltration mechanism.
Forensic Artifacts	GitHub Organization Audit Log (API: GET /orgs/{org}/audit-log) filtered for 'oauth_access', 'git.clone', 'repo.download', and 'org.add_member' events — the stolen VS Code OAuth token would appear here tied to the victim developer's identity but originating from attacker-controlled infrastructure, making source IP and user-agent the primary forensic discriminators. | VS Code renderer process logs at '%APPDATA%\Code\logs\' (Windows) or '~/.config/Code/logs/' (Linux/macOS) — the webview abuse mechanism routes an outbound HTTP request through VS Code's Electron renderer to exfiltrate the OAuth token, and these logs may capture the destination URL of the attacker's token-harvesting endpoint. | VS Code extension installation directories at '%USERPROFILE%\.vscode\extensions\' (Windows) or '~/.vscode/extensions/' (Linux/macOS) — any extension installed or modified around the time of the malicious link click should be treated as potentially malicious and preserved for static analysis of its package.json and compiled JavaScript for hardcoded C2 URLs or token exfiltration logic. | Host-level DNS query logs or endpoint DNS cache ('ipconfig /displaydns' on Windows, 'sudo dscacheutil -cachedump' on macOS) captured at time of incident — the webview-initiated token exfiltration request would resolve a non-Microsoft, non-GitHub domain, and the DNS query record may be the only network-layer artifact available on endpoints without full packet capture. | GitHub Developer Settings OAuth App authorization list (Settings > Applications > Authorized OAuth Apps) for each affected developer account — this records which OAuth applications held token grants at time of incident including grant date and scope, establishing the forensic baseline for what the attacker could have accessed with the stolen token before revocation.

Per-Action IR Details

Containment — Immediately restrict VS Code's ability to open untrusted external links: enforce the 'security.workspace.trust' setting and set 'extensions.autoUpdate' to false in enterprise VS Code policy. Revoke and rotate any GitHub OAuth tokens issued to developer accounts that use VS Code or github.dev, prioritizing accounts with access to sensitive or production repositories. This maps to NIST AC-17 (Remote Access restrictions) and D3-CRO (Credential Rotation).

NIST Phase: Containment

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

Controls: NIST AC-17 (Remote Access), NIST AC-6 (Least Privilege), NIST IR-4 (Incident Handling), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Deploy VS Code settings enforcement via Group Policy (Windows) or a managed preferences profile (macOS) pushing 'security.workspace.trust.enabled': true and 'extensions.autoUpdate': false to all developer workstations immediately. For token revocation without a PAM tool, script GitHub API calls: use 'curl -H "Authorization: Bearer " https://api.github.com/orgs//installations' to enumerate active OAuth app authorizations, then revoke each with a DELETE to '/applications//tokens/'. A 2-person team can complete org-wide enumeration and revocation in under an hour using this approach.

Evidence: Before revoking tokens, capture the full list of active GitHub OAuth token grants and their associated scopes via the GitHub Admin API endpoint GET /orgs/{org}/credential-authorizations — this preserves a forensic record of which tokens existed, what scopes they held, and when they were last used, which is critical for determining blast radius. Also export VS Code's 'argv.json' and the per-user extension storage directory ('%APPDATA%\Code\User\globalStorage' on Windows, '~/.config/Code/User/globalStorage' on Linux/macOS) to document the installed extension state at time of incident before any remediation alters it.

Detection — Query git provider audit logs for OAuth token usage from unexpected IP addresses, geographies, or user agents. In GitHub, navigate to Settings > Security log and filter for 'oauth_access' events. Review VS Code extension telemetry and installed extension lists on developer workstations for unrecognized extensions (CIS 2.1 — Software Inventory; D3-SFA — System File Analysis). Search endpoint logs for VS Code webview network traffic to non-Microsoft, non-GitHub domains. No CVE or specific event ID is available yet — behavioral indicators are the primary detection path.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), NIST SI-4 (System Monitoring), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 8.2 (Collect Audit Logs)

Compensating: On developer workstations, run: 'code --list-extensions > extensions_baseline.txt' on each machine and diff against a known-good list to surface unrecognized extensions that may have been installed via the malicious webview interaction. For network traffic, use Wireshark or tcpdump filtering on VS Code's renderer process PID to capture webview-originated HTTP/S connections to non-Microsoft/non-GitHub hosts: 'sudo tcpdump -i any -w vscode_webview.pcap -Z root "(host != 127.0.0.1) and (port 443 or port 80)"' while VS Code is running. For GitHub audit log queries without a SIEM, use the GitHub Audit Log API: 'curl -H "Authorization: Bearer " "https://api.github.com/orgs//audit-log?phrase=action:oauth_access&per_page=100"' and parse output for user_agent strings not matching known VS Code or browser patterns, or source IPs outside expected developer ranges.

Evidence: Pull GitHub organization audit logs specifically for 'oauth_access', 'repo.download', 'git.clone', and 'org.oauth_app_access_requested' events in the 30-day window preceding detection — the webview exploit would have exfiltrated the token silently, so the first visible evidence is typically an API call from an unexpected IP or user-agent string shortly after the developer clicked the malicious link. On the developer endpoint, collect VS Code's renderer process network logs from '%APPDATA%\Code\logs' (Windows) or '~/.config/Code/logs' (Linux/macOS), which may contain outbound webview request URLs revealing the attacker's token-harvesting endpoint.

Eradication — No vendor patch is available. As a compensating control, enforce extension allowlisting via VS Code's 'extensions.allowedExtensionIDs' policy setting and restrict webview API access where enterprise policy permits. Remove any unrecognized VS Code extensions from developer machines (CIS 2.3 — Address Unauthorized Software). Revoke all GitHub Personal Access Tokens and OAuth tokens for affected developers and reissue with minimum-necessary scopes per NIST AC-6 (Least Privilege). Apply D3-UAP (User Account Permissions) to limit GitHub token scope organization-wide.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-6 (Least Privilege), NIST CM-7 (Least Functionality), NIST SI-2 (Flaw Remediation), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Build the extension allowlist by running 'code --list-extensions' across all developer machines, consolidate into an approved set, and push 'extensions.allowedExtensionIDs' via enterprise policy (settings.json enforced at machine scope). For PAT/OAuth reissuance with scoped permissions, use GitHub's fine-grained PATs (available under Settings > Developer Settings > Fine-grained tokens) and limit new tokens to specific repository access rather than org-wide — document each token's justification in a shared tracking sheet as a compensating control for a PAM system. Remove unauthorized extensions via CLI: 'code --uninstall-extension ' run as a script across all machines.

Evidence: Before removing any extension, preserve a forensic copy of its installation directory under '~/.vscode/extensions//' or '%USERPROFILE%\.vscode\extensions\\' — a malicious or trojanized extension installed via the webview abuse vector would reside here and may contain obfuscated JavaScript payloads or hardcoded exfiltration endpoints worth analyzing. Additionally, export the VS Code 'keybindings.json' and 'settings.json' files per affected user, as some webview-based attacks modify workspace settings to establish persistence or re-enable auto-update.

Recovery — After token rotation, verify repository access audit logs show no anomalous clone, download, or settings-change events in the 30 days prior to detection. Re-enable only allow-listed VS Code extensions. Confirm enterprise VS Code policy changes have propagated to all developer workstations (NIST CM — Configuration Management baseline). Monitor GitHub organization audit logs for minimum 30 days post-remediation for residual token reuse (NIST AU-6 — Audit Record Review, Analysis, and Reporting; D3-LAM — Local Account Monitoring).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-2 (Baseline Configuration), NIST CM-6 (Configuration Settings), NIST IR-4 (Incident Handling), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure)

Compensating: Verify policy propagation by running a lightweight osquery query on developer workstations: 'SELECT key, value FROM vscode_settings WHERE key IN ("security.workspace.trust.enabled", "extensions.autoUpdate", "extensions.allowedExtensionIDs");' — or, absent osquery, collect settings.json from each machine path and grep for the enforced values in a shell loop. For GitHub audit log monitoring without a SIEM, schedule a daily cron job or GitHub Action that queries the Audit Log API for 'git.clone', 'repo.download', and 'oauth_access' events and emails a diff report to the security team.

Evidence: Before declaring recovery complete, pull the full GitHub organization audit log for the 30-day pre-detection window and specifically search for 'git.clone' and 'repo.download' events against private or production repositories from any IP or user-agent not matching known developer baselines — a stolen OAuth token used by an attacker would appear in this log tied to the original developer's identity, making scope determination dependent on this data. Retain these logs in an immutable store (e.g., exported to S3 with Object Lock, or appended to a write-once log file) to support potential breach notification analysis.

Post-Incident — This vulnerability exposed gaps in developer workstation extension governance (CIS 2.1, CIS 2.3) and OAuth token scoping discipline (NIST AC-6, CIS 6.1). Initiate a review of all GitHub OAuth token grants across the organization, enforce least-privilege token scoping as a standing policy, and add VS Code extension inventory to the quarterly asset review cycle (CIS 1.1). Consider whether github.dev access should be restricted via network policy until a vendor patch is available. Track the Microsoft VS Code security advisory page and CVE assignment for this issue; patch immediately upon release.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST AC-6 (Least Privilege), NIST SI-2 (Flaw Remediation), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 6.1 (Establish an Access Granting Process), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: To restrict github.dev access without enterprise proxy infrastructure, push a browser extension policy (e.g., via Chrome Enterprise or Firefox policies) that blocks navigation to 'github.dev' until a vendor patch is confirmed. For ongoing CVE/advisory tracking without a commercial feed, subscribe to the Microsoft Security Response Center (MSRC) RSS feed for VS Code advisories and set a GitHub watch on the 'microsoft/vscode' repository's Security Advisories tab — both are free and will surface a CVE assignment or patch release within hours of publication. Document the token scoping review findings in a lightweight risk register entry that ties back to this incident for future audit evidence.

Evidence: Produce a formal lessons-learned document that includes: (1) a timeline reconstructed from GitHub audit logs and VS Code logs mapping when the malicious link was clicked to when the token was first misused, (2) the full inventory of GitHub OAuth tokens that existed at time of incident with their scopes, and (3) a list of all private repositories accessible via any compromised token — this last item determines whether breach notification obligations apply under applicable regulations (e.g., GDPR, state breach notification laws) if source code containing PII or credentials was potentially accessed.

Detection Guidance

No CVE is assigned and no vendor signature exists, detection must rely on behavioral and log-based indicators. Primary detection paths: (1) GitHub audit log, filter oauth_access and repo.clone events for tokens used from IP addresses or user agents inconsistent with the developer's normal pattern; GitHub organization owners can access this at Settings > Audit log. (2) Endpoint extension inventory, enumerate installed VS Code extensions on all developer machines and flag any extension not on the approved allowlist (CIS 2.1); extensions installed silently will not match a user-initiated install record. (3) VS Code process network telemetry, look for outbound HTTP POST requests from the VS Code renderer process to non-Microsoft, non-GitHub domains, which would indicate token exfiltration. (4) GitHub token scope review, tokens with unexpectedly broad repo scope on accounts that should have narrow access are a secondary indicator. No specific hash IOCs, domains, or IPs are confirmed in available sources as of 2026-03-04. The Aikido.dev research post and BleepingComputer report are the primary technical references for updated IOCs, monitor both for additions. Confidence in behavioral detection: medium. IOC-based detection: low (no confirmed IOCs yet).

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.aikido.dev/blog/vs-code-extension-github-breach`	Aikido Security research post documenting the webview postMessage exploit chain and technical proof-of-concept detail — primary technical reference for detection and IOC updates	MEDIUM
URL	`https://www.bleepingcomputer.com/news/security/vs-code-zero-day-lets-hackers-steal-github-tokens-in-one-click/`	BleepingComputer report corroborating disclosure — secondary reference for updated IOCs and patch status tracking	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1059.007** — JavaScript
- **T1176** — Software Extensions
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1528** — Steal Application Access Token

- **T1204.001** — Malicious Link
- **T1550.001** — Application Access Token
- **T1566.002** — Spearphishing Link

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **AC-3** — Access Enforcement
- **IA-5** — Authenticator Management
- **SI-10** — Information Input Validation
- **SI-7** — Software, Firmware, and Information Integrity
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A03:2021** — Injection
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **5.2** — Use Unique Passwords
- **16.10** — Apply Secure Design Principles in Application Architectures
- **2.5** — Allowlist Authorized Software
- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets

HIPAA-SECURITY

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

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities

NIST-CSF-2

- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1059.007	JavaScript	Execution
T1176	Software Extensions	Persistence
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1528	Steal Application Access Token	Credential-Access
T1204.001	Malicious Link	Execution
T1550.001	Application Access Token	Defense-Evasion
T1566.002	Spearphishing Link	Initial-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/vs-code-zero-day-let...	T3
Security - microsoft/vscode - GitHub	https://github.com/microsoft/vscode/security	T3
Scam Alert: Fake "VS Code Critical Vulnerability" post mass-pinging ...	https://www.reddit.com/r/github/comments/1s3ksik/scam_alert_fake_vs...	T3
Security - Visual Studio Code	https://code.visualstudio.com/docs/copilot/security	T3
The Wild West of VS Code extensions and how a poisoned ...	https://www.aikido.dev/blog/vs-code-extension-github-breach	T3

Generated 2026-06-03 13:58 UTC by TJS Security Command Center