INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:36 UTC

GitHub Discussions Weaponized at Scale: Fake VS Code Alerts Funnel Developers Through Reconnaissance Pipeline

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0116
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	GitHub Discussions platform (notification system abused), Visual Studio Code extensions ecosystem (impersonated), Google Drive (payload staging); developer workstations across open-source and enterprise repositories
Published	2026-03-27
Discovery Source	Rss

Executive Summary

Attackers are injecting fabricated VS Code vulnerability alerts into GitHub Discussion threads at scale, exploiting GitHub's native notification system to reach developers directly. The campaign impersonates repository maintainers, cites non-existent CVE identifiers to manufacture urgency, and directs victims to Google Drive-hosted payloads that profile and selectively infect developer systems. Any organization with developers subscribed to GitHub repository notifications is exposed; successful compromise of a developer workstation creates direct access to source code, credentials, CI/CD pipelines, and internal systems.

Technical Analysis

This is a multi-stage, automated social engineering campaign targeting developers through GitHub's Discussion notification system. Attackers post fabricated vulnerability alerts in repository Discussion threads, impersonating maintainers and citing invented CVE identifiers. Victims who follow embedded links land on Google Drive-staged payloads. The first stage is a JavaScript-based reconnaissance implant that profiles the victim environment, fingerprinting OS, browser, and tooling, and filters out automated scanners, bots, and security researchers before deciding whether to proceed. A second-stage payload exists but had not been captured or characterized as of reporting; its capabilities are unknown. No legitimate CVEs are associated with this campaign. Relevant CWEs: CWE-345 (Insufficient Verification of Data Authenticity), CWE-1021 (Improper

Restriction of Rendered UI Layers), CWE-693 (Protection Mechanism Failure). MITRE ATT&CK coverage includes T1566.003 (Phishing: Spearphishing via Service), T1204.002 (User Execution: Malicious File), T1608.004 (Stage Capabilities: Drive-by Compromise), T1497.001 (Virtualization/Sandbox Evasion: System Checks), T1195/T1195.001/T1195.002 (Supply Chain Compromise variants), T1027/T1027.011 (Obfuscated Files or Information), T1583.001 (Acquire Infrastructure: Domains), T1071.001 (Application Layer Protocol: Web Protocols). Attribution is not established. Source base is community-reported and secondary news outlets; no T1 formal threat intelligence advisory issued as of reporting (Socket.dev primary research noted). Confidence in technical details is moderate pending independent vendor corroboration.

Action Checklist

1. Step 1: Containment. Alert all developers to treat unsolicited GitHub Discussion notifications citing VS Code vulnerabilities or urgent CVE IDs as suspect. Do not click embedded links before verification. Block Google Drive (drive.google.com and related storage endpoints) at the proxy or firewall for developer workstations if operationally feasible, or flag all Google Drive downloads from developer systems for immediate review. Notify your development teams and any open-source project maintainers your organization supports.
2. Step 2: Detection. Review proxy and DNS logs for developer workstations showing outbound connections to Google Drive (drive.google.com, docs.google.com) originating from VS Code processes or browser sessions within 10-60 minutes following GitHub notification interaction. Search email and GitHub notification logs for Discussion posts containing phrases such as 'critical vulnerability,' 'CVE-' combined with VS Code references, and links to drive.google.com. Query endpoint detection logs for JavaScript execution initiated outside standard development workflows, particularly scripts performing system profiling (OS version checks, process enumeration, environment variable reads) shortly after browser activity. Check GitHub audit logs for Discussion posts at abnormal volume or from accounts with no prior contribution history.
3. Step 3: Eradication. For any workstation where a Google Drive payload was downloaded and executed: isolate immediately. Conduct forensic triage focused on JavaScript execution artifacts, browser history, and process trees from the time of interaction. Remove any downloaded files from Google Drive. Revoke and rotate all credentials stored on or accessible from the affected workstation, including Git credentials, SSH keys, cloud provider tokens, and CI/CD secrets. Review GitHub personal access tokens and OAuth app authorizations for affected accounts and revoke anything unrecognized.
4. Step 4: Recovery. Before returning isolated workstations to service, confirm no persistence mechanisms are present (scheduled tasks, startup entries, modified shell profiles, browser extensions added without authorization). Rotate all secrets that were accessible on affected systems, including those in CI/CD pipelines, secret managers, and environment variable stores. Validate integrity of any code commits made from affected workstations in the period following the suspected interaction. Monitor outbound traffic from recovered workstations for 14 days for anomalous connections.
5. Step 5: Post-Incident. This campaign exploits the absence of CVE verification habits among developers. Implement a standing developer security awareness touchpoint: unsolicited security alerts via GitHub Discussions, issues, or comments citing CVEs should always be verified against the NVD (nvd.nist.gov) and the official project advisory page before any action is taken. Evaluate whether GitHub notification scope for external repositories can be narrowed. Assess whether Google Drive and similar consumer cloud storage services require access controls in developer environments. Document this incident pattern for future phishing simulation scenarios targeting development teams.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to executive leadership, legal, and (if applicable) your breach notification counsel immediately if forensic triage confirms the Google Drive JavaScript payload executed system profiling AND exfiltrated data, if any CI/CD pipeline secrets or cloud provider credentials were exposed, or if supply chain integrity of code committed from affected workstations cannot be confirmed — any of these conditions creates potential regulatory notification obligations (SOC 2, PCI DSS, state breach notification statutes) and third-party disclosure requirements to downstream software consumers.
Recovery Notes	Before any affected developer workstation is returned to production use, independently verify via Autoruns and manual shell profile inspection that no persistence mechanism was installed by the reconnaissance payload, and validate via git log review that no commits modifying CI/CD workflow files, dependency manifests (package.json, requirements.txt, go.mod), or secret references were pushed from the affected account during the exposure window. All rotated credentials — GitHub PATs, SSH keys, cloud provider tokens, and CI/CD pipeline secrets — should be treated as fully compromised and must not be reused. Conduct active outbound traffic monitoring on recovered workstations for a minimum of 14 days using Sysmon Event ID 3 logs, specifically watching for connections to Google Drive, Pastebin, Discord CDN, or other consumer platforms that are atypical for developer toolchain traffic and consistent with second-stage C2 staging patterns used in developer-targeted campaigns.
Forensic Artifacts	Browser download history SQLite databases (Chrome: %LOCALAPPDATA%\Google\Chrome\User Data\Default\History; Firefox: %APPDATA%\Mozilla\Firefox\Profiles*.default\downloads.sqlite) — query the 'downloads' table for entries referencing drive.google.com or *.googleusercontent.com URLs to confirm payload retrieval and establish precise download timestamps | Windows Prefetch files at C:\Windows\Prefetch\ for wscript.exe, cscript.exe, node.exe, or powershell.exe — these record last execution time and referenced file paths, confirming whether the Google Drive-hosted JavaScript payload was executed even if the file was subsequently deleted | Sysmon Event ID 1 (Process Create) logs showing parent-child process chains where chrome.exe, msedge.exe, or firefox.exe spawned wscript.exe, cscript.exe, node.exe, or powershell.exe — this chain is the primary execution artifact of a browser-delivered JavaScript reconnaissance payload | GitHub notification emails in raw .eml format with full SMTP headers, preserving the fabricated CVE identifier text, the impersonated sender display name, and the drive.google.com payload link — the CVE identifiers will be verifiably absent from NVD and serve as the primary indicator of the lure campaign | Git credential store files (~/.git-credentials on Linux/macOS, %APPDATA%\Git\credentials on Windows) and SSH key pairs in ~/.ssh/ captured before rotation, to document the scope of developer credential exposure and identify which repositories, cloud accounts, and CI/CD systems were accessible with the compromised credentials

Per-Action IR Details

Step 1: Containment — Alert all developers to treat unsolicited GitHub Discussion notifications citing VS Code vulnerabilities or urgent CVE IDs as suspect. Do not click embedded links before verification. Block Google Drive (drive.google.com and related storage endpoints) at the proxy or firewall for developer workstations if operationally feasible, or flag all Google Drive downloads from developer systems for immediate review. Notify your development teams and any open-source project maintainers your organization

supports.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST IR-6 (Incident Reporting), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: On Windows developer workstations without a managed proxy, push an emergency Group Policy or run a PowerShell script to add drive.google.com and *.googleusercontent.com to the Windows Hosts file pointing to 127.0.0.1: `Add-Content C:\Windows\System32\drivers\etc\hosts '127.0.0.1 drive.google.com'; Add-Content C:\Windows\System32\drivers\etc\hosts '127.0.0.1 docs.google.com'`. On Linux/macOS, distribute a shell one-liner via configuration management or direct SSH: `echo '127.0.0.1 drive.google.com' | sudo tee -a /etc/hosts`. Simultaneously, send a plain-text alert via internal Slack/Teams/email with a screenshot example of the fake VS Code CVE Discussion post format so developers recognize the specific lure.

Evidence: Before blocking, capture a full proxy or DNS log export showing any developer workstations that already resolved drive.google.com or *.googleusercontent.com in the 72 hours preceding the containment action — this establishes the pre-containment exposure window. Preserve any GitHub Discussion notification emails in raw .eml format, including full headers, that reference 'VS Code', 'CVE-', or 'critical vulnerability' to document the lure content and sender impersonation pattern. Do not delete or quarantine these emails until forensic copies are confirmed.

Step 2: Detection — Review proxy and DNS logs for developer workstations showing outbound connections to Google Drive (drive.google.com, docs.google.com) originating from VS Code processes or browser sessions immediately following GitHub notification interaction. Search email and GitHub notification logs for Discussion posts containing phrases such as 'critical vulnerability,' 'CVE-' combined with VS Code references, and links to drive.google.com. Query endpoint detection logs for JavaScript execution initiated outside standard development workflows, particularly scripts performing system profiling (OS version checks, process enumeration, environment variable reads) shortly after browser activity. Check GitHub audit logs for Discussion posts at abnormal volume or from accounts with no prior contribution history.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-3 (Content of Audit Records), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy or validate Sysmon is running with a config that captures Event ID 1 (Process Create), Event ID 3 (Network Connection), and Event ID 11 (FileCreate). Run the following PowerShell query to detect browser or VS Code processes making outbound connections to Google Drive: `Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 3 -and $_.Message -match 'drive.google.com'} | Select-Object TimeCreated, Message | Export-Csv C:\IR\gdrive_connections.csv`. For JavaScript execution detection, query Sysmon Event ID 1 for wscript.exe, cscript.exe, or node.exe spawned by browser processes (chrome.exe, msedge.exe, firefox.exe): `Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 1 -and $_.Message -match 'wscript|cscript|node' -and $_.Message -match 'chrome|edge|firefox'}`. For GitHub audit log review without a SIEM, use the GitHub REST API: `curl -H 'Authorization: token YOUR_PAT' 'https://api.github.com/orgs/YOUR_ORG/audit-log?phrase=discussion.created&per_page=100'` and filter results for accounts created within the past 30 days or with zero prior contributions. Apply the public Sigma rule for suspicious JavaScript execution from browser child processes (search Sigma repository for 'browser_spawns_script_interpreter') to Windows Event Logs using Chainsaw or Hayabusa on exported EVTX files.

Evidence: Capture Sysmon Event ID 1 logs showing process creation chains where a browser spawned wscript.exe, cscript.exe, powershell.exe, or node.exe — these would represent the payload execution path from the Google Drive-hosted JavaScript. Collect Sysmon Event ID 3 logs showing network connections to drive.google.com or *.googleusercontent.com from VS Code (Code.exe) or browser processes within 10 minutes of a GitHub notification email timestamp. Export GitHub notification delivery logs and any Discussion post content citing 'CVE-' combined with

'Visual Studio Code' or 'VS Code extension' — the fabricated CVE identifiers in these posts will not resolve in NVD and are a primary forensic indicator. On macOS developer systems, collect Unified Log entries via `log collect --last 72h --output ~/ir_unified.logarchive` and filter for JavaScriptCore or osascript invocations following browser activity.

Step 3: Eradication — For any workstation where a Google Drive payload was downloaded and executed: isolate immediately. Conduct forensic triage focused on JavaScript execution artifacts, browser history, and process trees from the time of interaction. Remove any downloaded files from Google Drive. Revoke and rotate all credentials stored on or accessible from the affected workstation, including Git credentials, SSH keys, cloud provider tokens, and CI/CD secrets. Review GitHub personal access tokens and OAuth app authorizations for affected accounts and revoke anything unrecognized.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Before wiping, acquire a forensic image or at minimum a triage collection using Velociraptor (free, open-source) with the Windows.KapeFiles.Targets artifact, or run KAPE with the !SANS_Triage target to capture browser artifacts, prefetch, Sysmon logs, and scheduled tasks in one pass. For credential revocation without a PAM tool: (1) Run `git credential reject` and delete ~/.gitconfig tokens; (2) Enumerate and delete SSH keys via `ls -la ~/.ssh/` and `cat ~/.ssh/authorized_keys`; (3) On the affected GitHub account, navigate to Settings > Developer Settings > Personal Access Tokens and revoke all tokens, then Settings > Applications > Authorized OAuth Apps and revoke unrecognized apps. For CI/CD secret rotation, query your pipeline configuration files for hardcoded tokens using truffleHog (free): `trufflehog filesystem /path/to/repo --only-verified`. Remove downloaded payload files by searching the user's Downloads folder and browser cache: on Windows, `Get-ChildItem -Path $env:USERPROFILE\Downloads,"$env:LOCALAPPDATA\Google\Chrome\User Data\Default\Cache" -Recurse -Include *.js,*.zip,*.exe | Export-Csv C:\IR\payload_candidates.csv`.

Evidence: Before isolation or reimaging, capture: (1) Browser download history from Chrome/Edge at `%LOCALAPPDATA%\Google\Chrome\User Data\Default\History` (SQLite — query the 'downloads' table for drive.google.com URLs) or Firefox at `%APPDATA%\Mozilla\Firefox\Profiles\*.default\downloads.sqlite`; (2) Windows Prefetch files (`C:\Windows\Prefetch\`) for evidence of wscript.exe, cscript.exe, node.exe, or powershell.exe execution correlated to the incident timeframe — these persist even if the payload file was deleted; (3) The JavaScript reconnaissance payload file itself from the Downloads folder or browser cache, preserved with hash (SHA-256) before removal — this payload will contain system profiling logic (navigator.userAgent checks, process enumeration, environment variable reads) that reveals attacker targeting criteria; (4) The contents of any newly created or modified scheduled tasks (`schtasks /query /fo LIST /v > C:\IR\scheduled_tasks.txt`) and startup registry keys (`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`) written after the interaction timestamp; (5) Git credential store contents (`%APPDATA%\Git\credentials` or `~/.git-credentials`) to document which tokens were exposed prior to rotation.

Step 4: Recovery — Before returning isolated workstations to service, confirm no persistence mechanisms are present (scheduled tasks, startup entries, modified shell profiles, browser extensions added without authorization). Rotate all secrets that were accessible on affected systems, including those in CI/CD pipelines, secret managers, and environment variable stores. Validate integrity of any code commits made from affected workstations in the period following the suspected interaction. Monitor outbound traffic from recovered workstations for 14 days for anomalous connections.

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-2 (Baseline Configuration), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For persistence verification without an EDR, run the free Autoruns (Sysinternals) tool and export results: `autorunsc.exe -a * -c -h -s > C:\IR\autoruns_baseline.csv` — compare against a known-clean baseline from another workstation. Check browser extensions manually in Chrome at `chrome://extensions/` and cross-reference against a pre-incident screenshot or IT-maintained approved extension list. For commit integrity validation, run `git log --all --author-date-order --since='INCIDENT_DATE' --format='%H %an %ae %ai %s'` on affected repositories and visually inspect for commits from the affected developer account that introduce unexpected dependencies, new outbound URLs, or changes to CI/CD workflow files (`.github/workflows/*.yml`) — these are high-value supply chain tampering targets. For 14-day outbound monitoring without a SIEM, deploy Sysmon Event ID 3 collection and forward logs to a central syslog server using nxlog (free), filtering for connections outside approved developer destinations.

Evidence: Before returning the workstation to service, document: (1) Shell profile integrity — capture hash of `~/.bashrc`, `~/.zshrc`, `~/.bash_profile` (Linux/macOS) or PowerShell profile at `$PROFILE` (Windows) and compare against a clean reference; (2) Browser extension list with version numbers and installation timestamps from `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Extensions\` — the attack's reconnaissance payload may have profiled the browser environment, and a follow-on stage could install a malicious extension for persistent credential harvesting; (3) A signed export of all GitHub commits, PRs, and workflow file changes made from the affected account during the exposure window, to establish a chain of custody for supply chain integrity review; (4) CI/CD pipeline secret access logs (GitHub Actions audit log, accessible via API: `curl -H 'Authorization: token PAT' 'https://api.github.com/orgs/ORG/audit-log?phrase=secrets.get'`) to determine if any pipeline secrets were read during the compromise window.

Step 5: Post-Incident — This campaign exploits the absence of CVE verification habits among developers. Implement a standing developer security awareness touchpoint: unsolicited security alerts via GitHub Discussions, issues, or comments citing CVEs should always be verified against the NVD (nvd.nist.gov) and the official project advisory page before any action is taken. Evaluate whether GitHub notification scope for external repositories can be narrowed. Assess whether Google Drive and similar consumer cloud storage services require access controls in developer environments. Document this incident pattern for future phishing simulation scenarios targeting development teams.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-2 (Incident Response Training), NIST IR-8 (Incident Response Plan), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Build a one-page developer quick-reference card: 'If you receive a GitHub notification citing a CVE for VS Code or any extension, go directly to https://nvd.nist.gov/vuln/search and search the CVE ID before clicking anything — if it does not appear in NVD, it is fabricated.' Distribute via internal wiki and pin to developer Slack/Teams channels. For GitHub notification scope reduction, document the steps for developers to navigate to GitHub Settings > Notifications > Subscriptions and unwatch repositories where they are not active contributors — reducing the attack surface for this notification-abuse vector. For phishing simulation, work with your awareness training vendor (or use GoPhish, free and open-source) to build a simulation campaign that replicates this exact lure format: a GitHub-style notification email with a fabricated CVE ID referencing VS Code, linking to a benign Google Drive file, to measure developer click-through rates before and after awareness training. Log simulation results as a metric for NIST IR-3 (Incident Response Testing) compliance evidence.

Evidence: Compile the following for the post-incident report and lessons-learned meeting: (1) The complete list of fabricated CVE identifiers used in the Discussion posts, confirmed as non-existent via NVD query, to establish attacker methodology documentation; (2) A timeline mapping GitHub notification delivery timestamps to DNS/proxy log hits on drive.google.com, to quantify the median time-to-click and inform future detection rule thresholds; (3) The JavaScript payload (or behavioral analysis of it) documenting exactly which system profiling operations were performed — OS version, installed software enumeration, environment variable reads — to inform what attacker targeting criteria were and whether your environment was flagged for second-stage delivery; (4) GitHub audit log export showing the Discussion-posting accounts, their creation dates, prior activity, and whether they impersonated named maintainers, for potential abuse reporting to GitHub Security via security@github.com.

Detection Guidance

Primary detection surface is proxy and DNS telemetry. Look for outbound connections to drive.google.com or googleapis.com storage endpoints originating from developer workstations within a short time window (10-60 minutes) after a GitHub notification is received or a GitHub.com page is loaded. Secondary surface: endpoint telemetry showing JavaScript or Node.js processes spawned from browser sessions performing environment fingerprinting, specifically reading OS build information, enumerating running processes, or checking for virtualization indicators. Behavioral indicator: a script that exits or stalls without network callback on sandboxed or VM-based analysis environments is consistent with the described sandbox evasion (T1497.001). Search GitHub Discussion activity across repositories your organization maintains or contributes to for posts containing 'VS Code,' 'critical,' and 'CVE' combinations from accounts with zero prior contribution history. No confirmed IOC hashes or C2 infrastructure are available as of reporting; the second-stage payload remains uncharacterized. Treat any Google Drive-hosted file downloaded via a GitHub Discussion link as high-confidence suspicious pending isolated sandbox and manual security team analysis. Note: the absence of published IOCs reflects the campaign's active sandbox evasion; static signature detection alone is insufficient.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`drive.google.com (payloads staged here per campaign reporting)`	Google Drive used for first-stage JavaScript payload staging; specific URLs not published as of reporting	MEDIUM
DOMAIN	`drive.google.com`	Payload delivery infrastructure; shared Google infrastructure — block or alert selectively based on context, not globally	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1608.004** — Drive-by Target
- **T1195** — Supply Chain Compromise
- **T1497.001** — System Checks
- **T1566.003** — Spearphishing via Service
- **T1195.002** — Compromise Software Supply Chain
- **T1071.001** — Web Protocols
- **T1608.001** — Upload Malware
- **T1497** — Virtualization/Sandbox Evasion
- **T1027** — Obfuscated Files or Information
- **T1195.001** — Compromise Software Dependencies and Development Tools

- **T1027.011** — Fileless Storage
- **T1204.002** — Malicious File
- **T1583.001** — Domains
- **T1566** — Phishing
- **T1608** — Stage Capabilities

NIST-800-53R5

- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-8** — Spam Protection

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5**
- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1608.004	Drive-by Target	Resource-Development
T1195	Supply Chain Compromise	Initial-Access
T1497.001	System Checks	Defense-Evasion
T1566.003	Spearphishing via Service	Initial-Access
T1195.002	Compromise Software Supply Chain	Initial-Access
T1071.001	Web Protocols	Command-And-Control

Technique ID	Technique Name	Tactic
T1608.001	Upload Malware	Resource-Development
T1497	Virtualization/Sandbox Evasion	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1027.011	Fileless Storage	Defense-Evasion
T1204.002	Malicious File	Execution
T1583.001	Domains	Resource-Development
T1566	Phishing	Initial-Access
T1608	Stage Capabilities	Resource-Development

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/fake-vs-code-alerts-...	T3
r/github - Scam Alert: Fake "VS Code Critical Vulnerability ...	https://www.reddit.com/r/github/comments/1s3ksik/scam_alert_fake_vs...	T3
Security Vulnerability Report: Continue VS Code Extension	https://github.com/continuedev/continue/issues/9025	T3
Widespread GitHub Campaign Uses Fake VS Code Security ...	https://socket.dev/blog/widespread-github-campaign-uses-fake-vs-cod...	T3
Major Vulnerabilities in this extension · Issue #270124	https://github.com/microsoft/vscode/issues/270124	T3

Generated 2026-03-29 18:36 UTC by TJS Security Command Center