INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-20 18:55 UTC

GitHub Breached via Poisoned VS Code Extension: TeamPCP Exfiltrates ~3,800 Internal Repositories in Active Supply Chain Campaign

DATA BREACH | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-DBR-2026-0135
Type	Data Breach
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	GitHub internal repositories (~3,800), Microsoft Visual Studio Code (malicious third-party extension); affected repositories include GitHub Actions, Copilot, CodeQL, Codespaces, Dependabot, and internal infrastructure
Published	2026-05-20T21:07:38+00:00
Discovery Source	Rss:T1 Psirt

Executive Summary

On May 18, 2026, threat actor TeamPCP compromised a GitHub employee device via a malicious Visual Studio Code extension, exfiltrating approximately 3,800 internal repositories containing source code for GitHub Actions, Copilot, CodeQL, Codespaces, Dependabot, and internal infrastructure. Internal repositories are suspected to contain customer support data, though the full scope of exposure has not been verified by GitHub. This breach exposes proprietary development tooling and AI-assisted coding infrastructure to a threat actor actively conducting broader supply chain operations, creating downstream risk for the millions of developers and enterprises that depend on GitHub's platform integrity.

Technical Analysis

TeamPCP gained initial access via a trojanized third-party VS Code extension installed on a GitHub employee device (MITRE T1176, Browser Extensions/IDE Extensions). The extension facilitated credential theft (CWE-522, T1552.001) and session cookie theft (T1539), providing persistent access to internal GitHub systems. Lateral movement occurred via Remote Services including SSH (T1021.001) and AWS SSM (T1021). Exfiltration used Web Service channels (T1567), with reported use of GitHub commit messages as C2. Compromised repositories include source for GitHub Actions, Copilot, CodeQL, Codespaces, and Dependabot, all critical components of the software supply chain. This incident is part of a broader campaign distributing malicious tooling via poisoned third-party extension and supply chain channels. Relevant CWEs: CWE-829

(Inclusion of Functionality from Untrusted Control Sphere), CWE-494 (Download of Code Without Integrity Check), CWE-693 (Protection Mechanism Failure), CWE-522 (Insufficiently Protected Credentials). GitHub has confirmed the breach, rotated critical secrets, and opened an investigation. Patch status: no software patch applicable, this is an intrusion requiring organizational and credential remediation.

Action Checklist

1. Step 1: Containment. Audit all VS Code extensions installed across developer workstations and CI/CD build environments. Remove any unverified or recently installed third-party extensions. Cross-reference against the CISA Supply Chain guidance and your approved software inventory (CIS 2.1, CIS 2.3). Rotate all GitHub personal access tokens, OAuth tokens, GitHub App credentials, and deploy keys held by any employee who uses VS Code with third-party extensions. Prioritize tokens with repository or Actions access.
2. Step 2: Detection. Review endpoint and identity logs for the following behavioral indicators: (a) VS Code extension processes spawning network connections to unexpected external domains; (b) credential or token access from unusual source IPs or outside normal working hours (NIST AU-6); (c) GitHub API calls at unusual volume or from new devices, particularly clone or archive operations against internal repositories (T1213); (d) SSH or SSM session establishment from unfamiliar hosts (T1021.001); (e) Python interpreter execution (T1059.006) spawned by IDE processes. Query EDR telemetry for extension host processes making outbound connections. Review GitHub audit logs for bulk repository access or cloning events around May 18, 2026 and the days prior.
3. Step 3: Eradication. Enforce an allowlist of approved VS Code extensions via policy (CIS 2.3, NIST CM-7). Remove all unapproved extensions from developer endpoints. Enforce extension signature verification and integrity checking where tooling supports it. Revoke and reissue all secrets, tokens, and certificates associated with affected repository systems (NIST SC-7). Audit GitHub Actions workflow files in any repository that may have been accessed post-compromise for injected malicious steps.
4. Step 4: Recovery. Validate that all rotated secrets are propagated to dependent systems without service disruption. Monitor GitHub Actions pipeline execution logs and CodeQL scan results for anomalous behavior indicating tool compromise. Re-baseline behavioral alerts for developer identity activity (NIST AU-6, NIST SI-4). Confirm no unauthorized changes were introduced to Actions, Copilot, or Dependabot codebases that could propagate downstream to end users. Review system file and configuration baselines for any modifications on affected developer machines.
5. Step 5: Post-Incident. Conduct a control gap review against NIST AC-6 (Least Privilege) and AC-3 (Access Enforcement) for developer workstation permissions and repository access scopes. Implement MFA enforcement for all GitHub enterprise access (CIS 6.3, CIS 6.5). Establish a formal VS Code extension governance process with a documented approved list, periodic review, and automated enforcement. Brief development teams on supply chain compromise tradecraft, specifically T1195.001 and T1176. Review PyPI package dependencies in CI/CD pipelines for indicators of malicious packages. Document lessons learned and update playbooks for IDE-based supply chain intrusion scenarios.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate immediately to legal counsel and executive leadership if forensic review of the ~3,800 exfiltrated repositories confirms the presence of customer PII, regulated data, or cryptographic signing keys for GitHub Actions or Dependabot release artifacts, as this triggers breach notification obligations under applicable data protection regulations and creates imminent downstream supply chain risk to all GitHub Actions and Dependabot consumers.
Recovery Notes	Recovery cannot be declared complete until git history integrity for the GitHub Actions, Copilot, CodeQL, Codespaces, and Dependabot codebases has been verified against pre-compromise baselines and all workflow YAML files audited for injected malicious steps, given that tampered CI/CD tooling could propagate the compromise to downstream GitHub users at scale. Maintain a 90-day elevated monitoring posture on developer identity activity, GitHub API usage patterns, and Actions pipeline execution logs, as TeamPCP may retain copies of exfiltrated credentials or codebase knowledge enabling a return intrusion after initial containment. Pay particular attention to Dependabot and GitHub Actions Marketplace artifact integrity throughout this window, as these are the highest-risk downstream propagation vectors from the exfiltrated repositories.
Forensic Artifacts	VS Code malicious extension package: full contents of `~/.vscode/extensions/{malicious-extension-id}/` including `package.json` (activation events, permissions), compiled JavaScript in `out/` or `dist/`, and any embedded or fetched Python scripts associated with the Mini Shai-Hulud infostealer component | GitHub enterprise audit log: events of type `git.clone`, `repo.download`, `repo.create_fork`, `oauth_access`, and `workflow_run` for the window May 1–18, 2026, filtered on developer accounts associated with VS Code workstations, to reconstruct the full exfiltration sequence of the ~3,800 repositories | Sysmon Event ID 3 (Network Connection) logs from affected developer workstations showing outbound connections originating from `extensionHost.exe` or `code.exe` to TeamPCP C2 infrastructure, including destination IPs, ports, and DNS names resolved at time of connection | Credential store artifacts from affected developer machines — macOS Keychain (`~/Library/Keychains/login.keychain-db`), Windows Credential Manager (`cmdkey /list` output), and Linux `~/.git-credentials` or `~/.netrc` files — representing the token harvest surface targeted by the Mini Shai-Hulud infostealer | Git commit and diff history for `.github/workflows/` directories across all repositories in the Actions, Copilot, CodeQL, Codespaces, and Dependabot product families, capturing any workflow YAML modifications made by compromised developer identities between initial compromise and containment as evidence of potential downstream supply chain tampering

Per-Action IR Details

Step 1: Containment — Audit all VS Code extensions installed across developer workstations and CI/CD build environments. Remove any unverified or recently installed third-party extensions. Cross-reference against the CISA Supply Chain guidance and your approved software inventory (CIS 2.1, CIS 2.3). Rotate all GitHub personal access tokens, OAuth tokens, GitHub App credentials, and deploy keys held by any employee who uses VS Code with third-party extensions. Prioritize tokens with repository or Actions access.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), NIST AC-2 (Account Management), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software)

Compensating: On each developer workstation, run: `code --list-extensions > extensions_$(hostname).txt` to inventory all installed VS Code extensions, then diff against a known-good baseline list. For CI/CD build agents (GitHub-hosted or self-hosted), inspect `.vscode/extensions.json` and the runner's `~/.vscode-server/extensions/` directory. Use osquery with `SELECT name, version, path FROM vscode_extensions;` if the osquery VS Code table is available, or parse the extensions directory manually with `ls -la ~/.vscode/extensions/`. To revoke GitHub tokens

without a PAM tool, use the GitHub REST API: `gh auth token` to identify active tokens per user, then `gh api -X DELETE /applications/{client_id}/token` for OAuth tokens, and audit deploy keys via `gh api /repos/{owner}/{repo}/keys` for each repository.

Evidence: BEFORE removing extensions, image or snapshot the VS Code extensions directory (`%APPDATA%\Code\User\extensions\` on Windows, `~/.vscode/extensions/` on Linux/macOS) to preserve the malicious extension package, its `package.json` manifest, and any embedded scripts. Capture the extension's activation events and `contributes` fields from `package.json` — the TeamPCP extension likely declared broad `onStartupFinished` or `*` activation to ensure execution. Preserve GitHub audit log exports (`https://github.com/organizations/{org}/settings/audit-log`) filtered to the window around May 18, 2026, covering `repo.clone`, `repo.download`, and `oauth_access` event types. Capture all active PAT and OAuth token records from GitHub enterprise admin console before rotation to establish the full blast radius.

Step 2: Detection — Review endpoint and identity logs for the following behavioral indicators: (a) VS Code extension processes spawning network connections to unexpected external domains; (b) credential or token access from unusual source IPs or outside normal working hours (NIST AU-6); (c) GitHub API calls at unusual volume or from new devices, particularly clone or archive operations against internal repositories (T1213); (d) SSH or SSM session establishment from unfamiliar hosts (T1021.001); (e) Python interpreter execution (T1059.006) spawned by IDE processes. Query EDR telemetry for extension host processes making outbound connections. Review GitHub audit logs for bulk repository access or cloning events around May 18, 2026 and the days prior.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST AU-3 (Content of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon with a configuration that logs Event ID 3 (Network Connection) filtered on parent process `code.exe` or `extensionHost.exe` — any outbound connection from the VS Code extension host to non-Microsoft, non-GitHub IP space is highly suspicious for the TeamPCP exfiltration channel. Use Sysmon Event ID 1 (Process Create) to detect Python interpreter (`python.exe`, `python3`) spawned with parent `extensionHost.exe`, matching MITRE T1059.006. For GitHub audit log analysis without a SIEM, download the audit log CSV from the GitHub enterprise admin panel and run: `grep -E 'repo.clone|repo.download|git.clone' audit.csv | awk -F',' '{print $2,$4,$6}' | sort | uniq -c | sort -rn` to surface high-volume cloning actors. Write a Sigma rule targeting Sysmon EID 3 with `Image|endswith: 'extensionHost.exe'` and `DestinationIp` not in the approved Microsoft/GitHub CIDR blocks.

Evidence: Capture Sysmon Network Connection logs (Event ID 3) for `extensionHost.exe` and `code.exe` processes for the 30 days prior to May 18, 2026 to establish C2 callback patterns for the TeamPCP infrastructure. Preserve GitHub enterprise audit logs in full — specifically `git.clone`, `repo.download`, `repo.create_fork`, and `oauth_access` events — filtering on actor accounts associated with VS Code developer workstations. Collect Windows Security Event Log Event ID 4688 (Process Creation) or Linux `/var/log/audit/audit.log` `execve` syscall records showing Python spawned by VS Code extension host processes, which would indicate the Mini Shai-Hulud infostealer component executing. Capture DNS query logs from developer workstations (Windows Event ID 22 via Sysmon or `/var/log/syslog` resolver logs) for domains queried by the extension host process to identify TeamPCP C2 domains.

Step 3: Eradication — Enforce an allowlist of approved VS Code extensions via policy (CIS 2.3, NIST CM-7). Remove all unapproved extensions from developer endpoints. Apply D3-FMBV (File Magic Byte Verification) and integrity checking to installed extension packages where tooling supports it. Enforce extension signature verification. Revoke and reissue all secrets, tokens, and certificates associated with affected repository systems (D3-CRO — Credential Rotation). Audit GitHub Actions workflow files in any repository that may have been modified post-compromise for injected malicious steps.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST CM-7 (Least Functionality), NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), NIST CM-3 (Configuration Change Control), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Enforce VS Code extension allowlisting without MDM by deploying a `settings.json` policy file with `extensions.autoUpdate: false` and a startup script that diffs the installed extensions directory against a signed approved-list manifest, alerting or removing unlisted entries. For GitHub Actions workflow integrity checking without enterprise tooling, use: `git log --all --diff-filter=M --name-only -- '.github/workflows/*.yml'` across all affected repositories to identify workflow files modified after the initial compromise date of May 18, 2026. Compute SHA-256 hashes of all `.vsix` extension packages found on developer machines (`sha256sum ~/.vscode/extensions/*/*.vsix`) and compare against VS Code Marketplace published checksums. Use YARA rules targeting the Mini Shai-Hulud infostealer's known Python bytecode patterns or string signatures against the extension directory if IOCs are available from threat intelligence feeds.

Evidence: Before uninstalling the malicious extension, preserve a full copy of its `.vsix` package and unpacked directory, including `extension/out/` compiled JavaScript, `package.json` activation hooks, and any bundled Python scripts — these constitute primary forensic evidence of the TeamPCP tooling. Export complete GitHub Actions workflow YAML files from all ~3,800 affected repositories at their current HEAD and compare against git history to detect injected malicious steps (e.g., steps that exfiltrate `GITHUB_TOKEN` or add unauthorized third-party actions). Capture the extension's network socket state at time of discovery if the process is still running: `ss -tnp | grep extensionHost` (Linux) or `netstat -anob | findstr extensionHost` (Windows). Preserve keychain/credential store artifacts on macOS (`~/Library/Keychains/`) or Windows Credential Manager (`cmdkey /list`) that the Mini Shai-Hulud infostealer may have harvested GitHub tokens from.

Step 4: Recovery — Validate that all rotated secrets are propagated to dependent systems without service disruption. Monitor GitHub Actions pipeline execution logs and CodeQL scan results for anomalous behavior indicating tampered tooling. Re-baseline behavioral alerts for developer identity activity (NIST AU-6, NIST SI-4). Confirm no unauthorized changes were introduced to Actions, Copilot, or Dependabot codebases that could propagate downstream to end users. Review D3-SFA (System File Analysis) outputs for any modified configuration or startup files on affected developer machines.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST CP-10 (System Recovery and Reconstitution), NIST CM-3 (Configuration Change Control), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without enterprise SIEM, establish a 30-day heightened monitoring window using GitHub's native audit log streaming (free tier) to a local syslog receiver, alerting on any `workflow_run` events for GitHub Actions pipelines in the Actions, Copilot, CodeQL, Codespaces, and Dependabot repositories that were among the ~3,800 exfiltrated. Run CodeQL's own CLI (`codeql database analyze`) against local clones of affected repositories to detect newly injected vulnerabilities or backdoors in the codebase — this is particularly critical for the Copilot and Dependabot codebases given downstream customer blast radius. For startup file integrity on developer machines, compute and record SHA-256 hashes of shell profiles (`~/.bashrc`, `~/.zshrc`, `~/.profile`), VS Code `settings.json`, and `tasks.json` files, then schedule a daily cron comparison: `sha256sum ~/.bashrc | diff - ~/.bashrc.baseline`.

Evidence: Before declaring recovery complete, audit git commit history across the Actions, Copilot, CodeQL, Codespaces, and Dependabot repositories for any commits signed by compromised developer identities between the estimated earliest compromise date and the date of containment — use `git log --author='' --after='2026-05-01' --before='2026-05-18'`. Capture GitHub Actions pipeline execution logs (available 90 days via API: `GET /repos/{owner}/{repo}/actions/runs/{run_id}/logs`) for all workflow runs in affected repositories to detect if tampered tooling produced anomalous artifacts or exfiltration steps during CI. Review Dependabot PR history for any auto-merged dependency updates during the compromise window that could have introduced a malicious package version.

Step 5: Post-Incident — Conduct a control gap review against NIST AC-6 (Least Privilege) and AC-3 (Access Enforcement) for developer workstation permissions and repository access scopes. Implement MFA

enforcement for all GitHub enterprise access (CIS 6.3, CIS 6.5, D3-MFA). Establish a formal VS Code extension governance process with a documented approved list, periodic review, and automated enforcement. Brief development teams on supply chain compromise tradecraft, specifically T1195.001 and T1176. Review PyPI package dependencies in CI/CD pipelines for indicators of the Mini Shai-Hulud infostealer campaign. Document lessons learned and update playbooks for IDE-based supply chain intrusion scenarios.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), NIST AC-3 (Access Enforcement), NIST IR-4 (Incident Handling), NIST RA-3 (Risk Assessment), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For PyPI dependency auditing without commercial SCA tooling, run `pip-audit` (free, maintained by PyPA) against all `requirements.txt` and `pyproject.toml` files in CI/CD pipeline definitions: `pip-audit -r requirements.txt --output json > audit_results.json`. Cross-reference installed package names and versions against the Mini Shai-Hulud campaign's known malicious PyPI package list using OSV (osv.dev) bulk query API at no cost. For extension governance enforcement without MDM, deploy a pre-commit hook and a CI/CD step that reads a signed `approved-extensions.json` manifest and fails the build if a developer's extension inventory (exported via `code --list-extensions`) contains unlisted entries. Develop a Sigma rule for the lessons-learned playbook targeting MITRE T1176 (Browser Extensions adapted for IDE extensions): process creation where `extensionHost.exe` or `code` spawns network-capable child processes outside approved domains.

Evidence: For the lessons-learned report, compile the full GitHub audit log export covering the 90-day window prior to May 18, 2026 to establish the true dwell time of the TeamPCP actor and whether earlier reconnaissance activity (T1213 — Data from Information Repositories) against internal repositories is detectable in hindsight. Preserve all forensic copies of the malicious VS Code extension, Mini Shai-Hulud Python components, and network IOCs (C2 domains, IPs, SSL certificate fingerprints) as reference artifacts for future detection rule development. Document the complete list of ~3,800 exfiltrated repository names and their classifications (source code, customer support data, infrastructure configs) to accurately scope downstream notification obligations and supply chain risk to consumers of GitHub Actions, Copilot, and Dependabot.

Detection Guidance

Primary detection focus: IDE extension process behavior, identity anomalies, and bulk repository access events. (1) Endpoint telemetry: alert on VS Code extension host processes (extensionHost.js, vscode-extension-host) initiating outbound network connections to domains not on an approved list, particularly over HTTPS to newly registered or low-reputation domains (T1583.001). (2) Credential access: alert on GitHub personal access tokens or OAuth tokens used from a new IP, new device fingerprint, or outside the user's normal geographic pattern within a short window (T1078). (3) Repository access volume: query GitHub audit logs for clone, archive-download, or GraphQL bulk query events against internal repositories exceeding baseline thresholds. Specifically flag access to repositories named or tagged as internal infrastructure, Actions, Copilot, CodeQL, or Dependabot. (4) Session cookie theft indicators: review IdP and GitHub SSO logs for session reuse from a second IP or user-agent immediately following a primary authentication event (T1539). (5) C2 via commit messages: alert on repository commit events with non-standard encoding patterns in commit messages, or commits from automated identities outside CI/CD service accounts (T1102). (6) Python execution: alert on python.exe or python3 processes spawned by IDE parent processes, particularly if they initiate network connections or access credential store locations (T1059.006). Recommended log sources: endpoint EDR telemetry, GitHub audit log stream, identity provider (IdP) access logs, network proxy/DNS logs for extension host processes.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`Not publicly disclosed at time of report`	TeamPCP C2 infrastructure associated with FIRESCALE C2 channel via GitHub commit messages — specific domains not confirmed in available sources	LOW
URL	`https://github.blog/security/investigating-unauthorized-access-to-githubs-internal-repositories/`	GitHub official incident disclosure — primary source for confirmed breach details and remediation actions taken by GitHub	HIGH

Framework Mappings

MITRE-ATTACK

- **T1213** — Data from Information Repositories
- **T1176** — Software Extensions
- **T1078** — Valid Accounts
- **T1021.001** — Remote Desktop Protocol
- **T1552.001** — Credentials In Files
- **T1102** — Web Service
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1555** — Credentials from Password Stores
- **T1583.001** — Domains
- **T1059.006** — Python
- **T1567** — Exfiltration Over Web Service
- **T1021** — Remote Services
- **T1041** — Exfiltration Over C2 Channel
- **T1539** — Steal Web Session Cookie
- **T1059** — Command and Scripting Interpreter
- **T1566** — Phishing

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AC-17** — Remote Access
- **AC-3** — Access Enforcement
- **CM-7** — Least Functionality

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **SI-3** — Malicious Code Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

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

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners
- **CC6.3** — Authorizes, modifies, or removes access

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1213	Data from Information Repositories	Collection

Technique ID	Technique Name	Tactic
T1176	Software Extensions	Persistence
T1078	Valid Accounts	Defense-Evasion
T1021.001	Remote Desktop Protocol	Lateral-Movement
T1552.001	Credentials In Files	Credential-Access
T1102	Web Service	Command-And-Control
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1555	Credentials from Password Stores	Credential-Access
T1583.001	Domains	Resource-Development
T1059.006	Python	Execution
T1567	Exfiltration Over Web Service	Exfiltration
T1021	Remote Services	Lateral-Movement
T1041	Exfiltration Over C2 Channel	Exfiltration
T1539	Steal Web Session Cookie	Credential-Access
T1059	Command and Scripting Interpreter	Execution
T1566	Phishing	Initial-Access

Sources

Source	URL	Tier
The latest security news for developers - The GitHub Blog	https://github.blog/security/investigating-unauthorized-access-to-g...	T3
	https://thehackernews.com/2026/05/github-investigating-teampcp-clai...	T3
	https://github.blog/security/investigating-unauthorized-access-to-g...	T3
	https://www.msn.com/en-us/news/technology/github-investigates-unaut...	T3
GitHub Breach Linked To Malicious VS Code Extension ... - LinkedIn	https://www.linkedin.com/pulse/github-breach-linked-malicious-vs-co...	T3

DISCLAIMER

Generated 2026-05-20 18:55 UTC by TJS Security Command Center