INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-26 18:35 UTC

Glassworm Botnet Takedown Exposes Developer Supply Chain as High-Value Attack Surface

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0366
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	VSCode, Cursor, Positron, Windsurf, VSCodium, OpenVSX marketplace, npm registry, PyPI, GitHub, Node.js environments, Windows, macOS, Linux
Discovery Source	Rss:T1 Threatintel

Executive Summary

On May 26, 2026, CrowdStrike, Google, and the Shadowserver Foundation disrupted Glassworm, a botnet that has targeted software developers since early 2025 by embedding malware in VSCode extensions, npm and PyPI packages, and GitHub repositories. The takedown severed command-and-control across all four channels, but any developer machine already infected with GlasswormRAT remains compromised until remediated. Organizations employing software developers face direct risk of source code theft, credential harvesting, and downstream supply-chain compromise affecting products, customers, and partners.

Technical Analysis

Glassworm is a multi-platform botnet (Windows, macOS, Linux) that has operated since at least early 2025, distributing GlasswormRAT through three primary vectors: trojanized VSCode extensions published to the OpenVSX marketplace and compatible IDEs (VSCode, Cursor, Positron, Windsurf, VSCodium), poisoned packages on npm and PyPI, and backdoored GitHub repositories. The campaign maps to CWE-506 (Embedded Malicious Code), CWE-494 (Download of Code Without Integrity Check), CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), and CWE-912 (Hidden Functionality). No CVE is assigned; the threat is campaign-based. MITRE ATT&CK techniques include T1195.001 and T1195.002 (Supply Chain Compromise), T1608.001 (Stage Capabilities: Upload Malware), T1546.011 (Application Shimming), T1059/T1059.006/T1059.007 (Command and Scripting Interpreter: Python, JavaScript), T1568/T1568.001 (Dynamic Resolution), T1219 (Remote Access Software), T1552.001 (Credentials from Files), T1547 (Boot or Logon Autostart Execution), T1505.003 (Web Shell), T1102/T1102.003 (Web Service), and T1071.001 (Application Layer Protocol: Web Protocols). The RAT provides persistent remote access, credential harvesting from developer environments, and a foothold for downstream supply-chain injection. The four C2 channels were

severed on May 26, 2026, but implants on already-infected machines remain active until manually eradicated. No vendor patch applies; remediation requires audit and removal of malicious extensions and packages.

Action Checklist

1. Step 1: Containment. Immediately suspend developer workstations showing anomalous outbound connections, unexpected process execution from IDE processes (code.exe, cursor.exe, node.exe), or unauthorized registry or startup modifications. Isolate affected machines from the CI/CD network segment. Audit all installed VSCode-family extensions against known-good baselines per CIS 2.1 (Establish and Maintain a Software Inventory) and revoke any developer tokens, API keys, or SSH keys stored in IDE credential stores on potentially compromised hosts (NIST AC-2, NIST IA controls).
2. Step 2: Detection. Query EDR and SIEM for process trees spawned by IDE extensions (e.g., code.exe or node.exe spawning unexpected child processes), outbound connections to non-standard domains from development environments, and modifications to npm/PyPI dependency lock files not tied to a code commit (NIST AU-6, AU-12). Hunt for persistence via T1547 indicators: new autostart entries, login items, or systemd units created by IDE or Node.js processes. Search for Python or JavaScript interpreters (T1059.006, T1059.007) executing from extension install directories. Review CI/CD pipeline logs for unexpected package pulls or build steps (CIS 8.2). Analyze system file integrity on extension directories and package manifests for embedded malicious code. Apply local account monitoring to detect credential access consistent with T1552.001.
3. Step 3: Eradication. Remove all VSCode-family extensions not explicitly approved and verified against publisher hash/signature. Purge node_modules directories and regenerate lock files from trusted, pinned package versions. Re-provision developer machines confirmed or suspected to be compromised rather than attempting in-place cleanup of GlasswormRAT, given its persistence mechanisms (T1547, T1505.003). Enforce CWE-494 mitigations: require integrity verification (checksum or signature) for all package installs. Block installation of extensions from OpenVSX and other registries that cannot verify publisher identity until registry-level remediation is confirmed. Rotate all credentials accessible from affected developer environments.
4. Step 4: Recovery. Before returning workstations to production, validate that no autostart persistence (T1547) remains, that all approved extensions match known-good hashes, and that outbound network behavior is clean. Re-scan all packages introduced to production builds during the campaign window (early 2025 through May 26, 2026) using software composition analysis tooling. Monitor CI/CD pipeline outputs for anomalous artifacts for a minimum of 30 days post-remediation (NIST SI-4, AU-6). Confirm least-privilege access for developer service accounts in build pipelines (NIST AC-6).
5. Step 5: Post-Incident. This campaign exposes four control gaps: absence of extension allowlisting (CIS 2.3, Address Unauthorized Software), lack of integrity checking on package downloads (CWE-494, NIST CM controls), insufficient monitoring of developer endpoint process execution (NIST AU-2, AU-12), and over-permissioned developer service accounts with access to production secrets (NIST AC-6, AC-5). Formalize an approved extension registry with hash verification. Enforce dependency pinning and lock file integrity checks in CI/CD gates. Implement behavioral monitoring on developer workstations as a separate detection tier. Review and reduce developer pipeline credentials to least-privilege. Conduct a supply-chain risk review per NIST SP 800-161r1 for any packages or build artifacts produced during the campaign window.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal, and external IR retainer immediately if forensic analysis confirms that CI/CD pipeline service accounts with access to production secrets (cloud provider keys, signing certificates, database credentials, customer PII repositories) were present on any GlasswormRAT-infected developer workstation, as this triggers mandatory breach notification assessment under GDPR Article 33, US state breach notification statutes, and SEC incident disclosure rules for public companies.
Recovery Notes	Before any compromised developer workstation returns to CI/CD network access, require a clean Autoruns export, a verified extension hash manifest signed by a second responder, and a 24-hour monitored network session with Sysmon Event ID 3 (Network Connect) logging active to confirm no residual C2 beaconing to former Glassworm infrastructure. All packages and build artifacts produced between early 2025 and May 26, 2026 must be treated as potentially tainted and re-scanned with SCA tooling before any downstream deployment or distribution to customers. Maintain elevated logging and behavioral alerting on developer endpoints and CI/CD pipeline jobs for a minimum of 30 days post-remediation, as GlasswormRAT's T1547 and T1505.003 persistence mechanisms may survive partial cleanup attempts on hosts that were not fully reimaged.
Forensic Artifacts	VSCode/Cursor/Windsurf/VSCodium extension directories ('%USERPROFILE%\.vscode\extensions\', '~/.vscode/extensions/', and IDE-specific equivalents) — GlasswormRAT was delivered as a malicious extension payload; these directories will contain the dropper code, obfuscated JavaScript, and any secondary payloads fetched from C2 infrastructure, with file timestamps indicating initial compromise date | npm cache and install logs ('~/.npm/_logs/', '%APPDATA%\npm-cache_logs\') and PyPI pip install history from shell history files — will document the malicious package names, versions, and registry endpoints contacted during initial infection via npm or PyPI supply-chain vector, establishing which packages need SCA re-review | Sysmon Event ID 1 (Process Create) logs filtered on ParentImage matching code.exe, cursor.exe, or node.exe with child processes being cmd.exe, powershell.exe, python.exe, or sh — the specific process tree GlasswormRAT creates when its extension-based loader executes, providing the forensic chain from IDE process to malicious execution | Windows Registry autorun keys (HKCU/HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, \RunOnce) and macOS LaunchAgents ('~/Library/LaunchAgents/', '/Library/LaunchDaemons/') and Linux systemd user units ('~/.config/systemd/user/') — GlasswormRAT's T1547 persistence entries will appear here as entries referencing node.exe, python.exe, or a renamed executable in the extension install directory | CI/CD pipeline build logs (GitHub Actions workflow run logs, Jenkins build console output, GitLab CI job logs) for all builds executed between early 2025 and May 26, 2026 — will reveal whether compromised package versions or malicious postinstall scripts from npm/PyPI were executed in the build environment, and whether any pipeline secrets (SIGNING_KEY, AWS_ACCESS_KEY_ID, DOCKER_PASSWORD) were potentially exfiltrated during build execution

Per-Action IR Details

Step 1: Containment — Immediately suspend developer workstations showing anomalous outbound connections, unexpected process execution from IDE processes (code.exe, cursor.exe, node.exe), or unauthorized registry or startup modifications. Isolate affected machines from the CI/CD network segment. Audit all installed VSCode-family extensions against known-good baselines per CIS 2.1 (Establish and

Maintain a Software Inventory) and revoke any developer tokens, API keys, or SSH keys stored in IDE credential stores on potentially compromised hosts (NIST AC-2, NIST IA controls).

NIST Phase: Containment

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

Controls: NIST AC-2 (Account Management), NIST AC-17 (Remote Access), NIST IA-4 (Identifier Management), NIST IR-4 (Incident Handling), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: On Windows: run 'Get-Process | Where-Object {$_.Parent.Name -in @("code","cursor","node")} | Select-Object Name,Id,Path,StartTime' to surface unexpected child processes from IDE parents. On macOS/Linux: 'ps auxf | grep -A5 -E "code|cursor|node"' for process trees. Enumerate installed VSCode extensions via 'code --list-extensions' and diff against a known-good text baseline stored in version control. To block CI/CD network access, apply host-based firewall rules: Windows — 'New-NetFirewallRule -DisplayName "Block CI Segment" -Direction Outbound -RemoteAddress -Action Block'; Linux — 'iptables -A OUTPUT -d -j DROP'. Revoke GitHub tokens via GitHub CLI: 'gh auth logout' and rotate npm tokens via 'npm token revoke '.

Evidence: Before isolating, capture: (1) full process tree snapshot including parent-child relationships for code.exe, cursor.exe, and node.exe using Sysinternals ProcDump or 'Get-CimInstance Win32_Process | Select-Object ProcessId,ParentProcessId,Name,CommandLine | Export-Csv'; (2) active network connections from IDE processes via 'netstat -anob > netstat_snapshot.txt' (Windows) or 'ss -antp > netstat_snapshot.txt' (Linux) to document C2 beacon destinations before network isolation; (3) VSCode extension directory listing with file hashes from '%USERPROFILE%\.vscode\extensions\' (Windows), '~/.vscode/extensions/' (macOS/Linux), and Cursor/Windsurf/VSCodium equivalents; (4) contents of IDE credential stores — VSCode SecretStorage backed by OS keychain — and any plaintext tokens in '~/.npmrc', '~/.pypirc', or environment variable exports visible in shell history; (5) Windows registry snapshot of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run for GlasswormRAT autostart entries.

Step 2: Detection — Query EDR and SIEM for process trees spawned by IDE extensions (e.g., code.exe or node.exe spawning unexpected child processes), outbound connections to non-standard domains from development environments, and modifications to npm/PyPI dependency lock files not tied to a code commit (NIST AU-6, AU-12). Hunt for persistence via T1547 indicators: new autostart entries, login items, or systemd units created by IDE or Node.js processes. Search for Python or JavaScript interpreters (T1059.006, T1059.007) executing from extension install directories. Review CI/CD pipeline logs for unexpected package pulls or build steps (CIS 8.2). Apply D3-SFA (System File Analysis) to inspect extension directories and package manifests for embedded malicious code. Apply D3-LAM (Local Account Monitoring) to detect credential access consistent with T1552.001.

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-12 (Audit Record Generation), NIST AU-2 (Event Logging), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with a community config (e.g., SwiftOnSecurity or Olaf Hartong's modular config) and tune Rule Groups to log Event ID 1 (Process Create) filtering on ParentImage containing 'code.exe', 'cursor.exe', or 'node.exe' with Image NOT matching expected node/npm binaries. Use Sigma rule 'proc_creation_win_susp_child_process_of_ide.yml' (write this rule if not present) to detect python.exe or cmd.exe spawned from extension directories. For npm lock file drift, run 'git diff HEAD package-lock.json yarn.lock' in all repos cloned during the campaign window and flag any dependency additions not present in commit history. Use 'osquery' query: 'SELECT name, path, start_time FROM processes WHERE parent IN (SELECT pid FROM processes WHERE name IN ("code","node","cursor"));' For systemd persistence: 'find /etc/systemd /home/*/.config/systemd -name "*.service" -newer /var/log/dpkg.log' to surface units created after IDE install dates.

Evidence: Before running detection queries, preserve: (1) Sysmon Event ID 1 logs from '%SystemRoot%\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx' covering the full campaign

window (early 2025 through May 26, 2026) — GlasswormRAT's extension-based loader will show node.exe or python.exe spawning from paths under '%USERPROFILE%\.vscode\extensions\\'; (2) npm audit and install logs from '%APPDATA%\npm-cache_logs\' (Windows) or '~/.npm/_logs/' (macOS/Linux) showing package resolution events and registry endpoints contacted; (3) PyPI install history from pip's 'pip list --format=json' output and any 'pip install' entries in shell history files (~/.bash_history, ~/.zsh_history); (4) CI/CD build logs from GitHub Actions, Jenkins, or GitLab CI — specifically steps invoking 'npm install', 'pip install', or 'npx' during builds triggered between early 2025 and May 26, 2026; (5) Windows Security Event Log Event ID 4663 (Object Access) on extension directories and Event ID 4688 (Process Creation) for python.exe/node.exe with command lines showing execution from extension install paths.

Step 3: Eradication — Remove all VSCode-family extensions not explicitly approved and verified against publisher hash/signature. Purge node_modules directories and regenerate lock files from trusted, pinned package versions. Re-provision developer machines confirmed or suspected to be compromised rather than attempting in-place cleanup of GlasswormRAT, given its persistence mechanisms (T1547, T1505.003). Enforce CWE-494 mitigations: require integrity verification (checksum or signature) for all package installs. Block installation of extensions from OpenVSX and other registries that cannot verify publisher identity until registry-level remediation is confirmed. Apply D3-CH (Credential Hardening) and D3-CRO (Credential Rotation) to rotate all credentials accessible from affected developer environments.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST CM-7 (Least Functionality), NIST CM-11 (User-Installed Software), NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), NIST AC-2 (Account Management), CIS 2.3 (Address Unauthorized Software), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: For extension removal, use 'code --uninstall-extension <publisher.extension_name>' for each non-approved extension, then verify the extension directory is empty with 'ls -la ~/.vscode/extensions/' and re-hash remaining entries: 'find ~/.vscode/extensions -name "package.json" -exec sha256sum {} \;' compared to a known-good manifest. For node_modules purge: 'find . -name "node_modules" -type d -prune -exec rm -rf {} +' followed by 'npm ci --ignore-scripts' (the '--ignore-scripts' flag prevents malicious postinstall hooks from re-executing during reinstall). Block OpenVSX at the DNS or firewall layer: add 'open-vsx.org' to a hosts-file blocklist or iptables OUTPUT DROP rule during the remediation window. For credential rotation without a PAM tool: revoke GitHub PATs at github.com/settings/tokens, rotate npm tokens via 'npm token revoke', and re-generate SSH keys with 'ssh-keygen -t ed25519' replacing any keys whose private copy existed on a compromised host.

Evidence: Before wiping and reprovisioning, image the compromised disk for forensic preservation per NIST 800-61r3 §3.4 guidance on evidence retention: use 'dd if=/dev/sdX | gzip > developer_host_$(hostname)_$(date +%Y%m%d).img.gz' or FTK Imager on Windows. Preserve: (1) the full VSCode extension directory including hidden files and any '.vsix' installer files cached locally — GlasswormRAT components delivered as malicious extension payloads will be present here; (2) any T1505.003 (Server Software Component) artifacts if the developer host ran a local dev server — check for unexpected entries in '%APPDATA%\npm\node_modules' or globally installed npm packages via 'npm list -g --depth=0'; (3) memory dump of node.exe and python.exe processes before termination using ProcDump: 'procdump -ma node_memdump.dmp' to capture in-memory C2 configuration or injected shellcode; (4) shell history files (~/.bash_history, ~/.zsh_history, ~/.node_repl_history) documenting commands run by the compromised extension; (5) macOS LaunchAgents/LaunchDaemons at '~/Library/LaunchAgents/' and '/Library/LaunchDaemons/' for GlasswormRAT persistence plists targeting macOS developer machines.

Step 4: Recovery — Before returning workstations to production, validate that no autostart persistence (T1547) remains, that all approved extensions match known-good hashes, and that outbound network behavior is clean. Re-scan all packages introduced to production builds during the campaign window (early 2025 through May 26, 2026) using software composition analysis tooling. Monitor CI/CD pipeline outputs for anomalous artifacts for a minimum of 30 days post-remediation (NIST SI-4, AU-6). Apply D3-UAP (User Account Permissions) to confirm least-privilege access for developer service accounts in build pipelines (NIST AC-6).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AC-6 (Least Privilege), NIST CM-3 (Configuration Change Control), NIST CP-10 (System Recovery and Reconstitution), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Autostart persistence validation on Windows: 'Get-CimInstance Win32_StartupCommand | Select-Object Name,Command,Location | Export-Csv startup_check.csv'; also check 'reg query HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' and 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run'. On Linux/macOS: 'systemctl list-units --type=service --state=enabled' and 'ls -la /etc/cron.d/ ~/Library/LaunchAgents/'. For SCA without commercial tooling, run 'npm audit --audit-level=high' and 'pip-audit' against all requirements.txt and package.json files from the campaign window. Hash-verify approved extensions: 'find ~/.vscode/extensions -maxdepth 2 -name "package.json" | xargs sha256sum | tee post_recovery_extension_hashes.txt' and diff against the pre-approved baseline. For CI/CD service account review: enumerate GitHub Actions secrets access with 'gh secret list' and confirm no production secrets (AWS_ACCESS_KEY, DATABASE_URL, SIGNING_KEY) remain accessible to build jobs that do not require them.

Evidence: During recovery validation, capture and retain: (1) a final Sysinternals Autoruns scan exported to XML — 'autorunsc.exe -a * -c -h -s > autoruns_post_recovery.csv' — as the authoritative record that no GlasswormRAT T1547 persistence survived reimaging; (2) network baseline capture using Wireshark or 'tcpdump -i any -w post_recovery_netflow_$(date +%Y%m%d).pcap' during the first developer session on the restored workstation to confirm absence of beaconing to former C2 infrastructure; (3) SCA tool output (npm audit, pip-audit, or OWASP Dependency-Check HTML report) for all packages in production artifacts built between early 2025 and May 26, 2026, retained as audit evidence; (4) CI/CD pipeline run logs for the 30-day monitoring window showing clean artifact hashes, serving as evidence that no compromised dependencies persisted into post-remediation builds; (5) IAM permission review output for all developer service accounts — GitHub Actions OIDC role policies, AWS IAM 'get-policy', or equivalent — confirming least-privilege configuration post-credential rotation.

Step 5: Post-Incident — This campaign exposes four control gaps: absence of extension allowlisting (CIS 2.3 — Address Unauthorized Software), lack of integrity checking on package downloads (CWE-494, NIST CM controls), insufficient monitoring of developer endpoint process execution (NIST AU-2, AU-12), and over-permissioned developer service accounts with access to production secrets (NIST AC-6, AC-5). Formalize an approved extension registry with hash verification. Enforce dependency pinning and lock file integrity checks in CI/CD gates. Implement behavioral monitoring on developer workstations as a separate detection tier. Review and reduce developer pipeline credentials to least-privilege. Conduct a supply-chain risk review per NIST SP 800-161r1 for any packages or build artifacts produced during the campaign window.

NIST Phase: Post Incident

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

Controls: NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), NIST AC-5 (Separation of Duties), NIST AC-6 (Least Privilege), NIST CM-7 (Least Functionality), NIST CM-11 (User-Installed Software), NIST RA-3 (Risk Assessment), NIST SA-12 (Supply Chain Protection), CIS 2.3 (Address Unauthorized Software), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Formalize extension allowlisting by committing an approved 'extensions.json' (VSCode recommended extensions file) to each project repository and deploying a pre-commit hook or CI gate that runs 'code --list-extensions | sort > installed.txt && diff approved_extensions.txt installed.txt && exit $?' to fail builds with unapproved extensions present. Enforce dependency pinning via 'npm ci' (requires exact package-lock.json) instead of 'npm install' in all CI pipelines, and add 'pip install --require-hashes -r requirements.txt' for Python projects (use 'pip-compile --generate-hashes' to build the hashes file). For behavioral monitoring on developer workstations with no EDR budget, deploy Sysmon with Florian Roth's recommended config and forward Event IDs 1, 3, 7, 11, 13 to a centralized syslog server (rsyslog or a free Graylog instance). For supply-chain artifact review, run OWASP Dependency-Check CLI ('dependency-check.sh --project GlasswormReview --scan ./artifacts/ --out ./report/') against all build outputs from the

campaign window.

Evidence: Retain for lessons-learned and potential regulatory reporting: (1) the complete incident timeline reconstructed from Sysmon, shell history, and CI/CD logs documenting which developer machines were affected, which extensions were present, and the earliest evidence of compromise — this establishes the regulatory notification clock for any jurisdictions requiring breach notification if production credentials or customer data were accessible from compromised pipelines; (2) the diff of package-lock.json and requirements.txt files between pre-campaign baselines and post-campaign states for all repos, demonstrating whether any malicious packages reached production builds; (3) the IAM/secrets audit output showing what production credentials were accessible from compromised developer environments — if AWS, GCP, or Azure production keys were present, escalate to a cloud forensics review of API call logs (CloudTrail, Cloud Audit Logs) for the campaign window; (4) the approved extension registry baseline established post-incident, version-controlled and signed, as the authoritative record for future deviation detection; (5) the NIST SP 800-161r1 supply-chain risk review report covering all third-party packages and build artifacts produced during the Glassworm campaign window, retained as evidence of due diligence for any customer, auditor, or regulatory inquiry.

Detection Guidance

Primary behavioral indicators: IDE processes (code.exe, cursor, node.exe) spawning unexpected shells or interpreters; outbound DNS or HTTP from extension processes to non-whitelisted domains (T1071.001, T1102); new autostart entries or startup scripts created by Node.js or Python processes (T1547); modification of package-lock.json, requirements.txt, or pyproject.toml files outside of committed code changes (T1195.001, T1195.002). File-based indicators: inspect VSCode-family extension directories for obfuscated JavaScript or Python payloads; verify file integrity on .vsix extension files to detect content mismatches. Log sources to query: EDR process execution logs (parent-child chain from IDE binaries), network proxy/firewall logs for outbound connections from developer subnet, npm/PyPI audit logs, CI/CD build logs for unexpected dependency additions, and system startup configuration. NIST AU-6 review cadence should be increased to daily for developer environments until the environment is validated clean. CIS 8.2 compliance (audit log collection) must be confirmed active on all developer workstations before hunting. Operationalization of specific IOC values (hashes, domains, IPs) requires cross-reference with the CrowdStrike primary source listed in references and human validation of indicators before deployment to detection tooling.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`[GlasswormRAT C2 domains — not confirmed from verified source in this session]`	Command-and-control infrastructure used by GlasswormRAT; specific values require human validation against the CrowdStrike primary source before operationalizing	LOW
HASH	`[GlasswormRAT payload hashes — not confirmed from verified source in this session]`	Malicious VSCode extension and package payload hashes; specific values require human validation against the CrowdStrike primary source	LOW

Framework Mappings

MITRE-ATTACK

- **T1568** — Dynamic Resolution
- **T1195.002** — Compromise Software Supply Chain
- **T1546.011** — Application Shimming
- **T1059.006** — Python
- **T1219** — Remote Access Tools
- **T1552.001** — Credentials In Files
- **T1608.001** — Upload Malware
- **T1059** — Command and Scripting Interpreter
- **T1078** — Valid Accounts
- **T1547** — Boot or Logon Autostart Execution
- **T1204.002** — Malicious File
- **T1102** — Web Service
- **T1071.001** — Web Protocols
- **T1059.007** — JavaScript
- **T1102.003** — One-Way Communication
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1568.001** — Fast Flux DNS
- **T1505.003** — Web Shell

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CM-2** — Baseline Configuration
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries

- **15.1** — Establish and Maintain an Inventory of Service Providers

NIST-CSF-2

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

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain

SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1568	Dynamic Resolution	Command-And-Control
T1195.002	Compromise Software Supply Chain	Initial-Access
T1546.011	Application Shimming	Privilege-Escalation
T1059.006	Python	Execution
T1219	Remote Access Tools	Command-And-Control
T1552.001	Credentials In Files	Credential-Access
T1608.001	Upload Malware	Resource-Development
T1059	Command and Scripting Interpreter	Execution
T1078	Valid Accounts	Defense-Evasion
T1547	Boot or Logon Autostart Execution	Persistence
T1204.002	Malicious File	Execution
T1102	Web Service	Command-And-Control
T1071.001	Web Protocols	Command-And-Control
T1059.007	JavaScript	Execution
T1102.003	One-Way Communication	Command-And-Control
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1568.001	Fast Flux DNS	Command-And-Control
T1505.003	Web Shell	Persistence

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/inside-crowdstrike-takedown-...	T3
Critical Open VSX Registry Flaw Exposes Millions of Developers to ...	https://thehackernews.com/2025/06/critical-open-vsx-registry-flaw-e...	T3
Malicious VS Code Extensions: Protect Developer Workstations	https://wizardcyber.com/malicious-vscode-extensions-threat/	T3
VSCodium - GitHub	https://github.com/VSCodium/vscodium	T3
Inside CrowdStrike's Takedown of a Developer-Targeting Botnet	https://www.crowdstrike.com/content/crowdstrike-www/locale-sites/us...	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-05-26 18:35 UTC by TJS Security Command Center