INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-24 06:19 UTC

Laravel Lang Supply Chain Compromise: Tag-Rewriting Attack Delivers Cross-Platform Credential Stealer to Developer Environments

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0360
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, laravel-lang/actions (Composer/Packagist); Chrome, Brave, Edge browsers (Windows); AWS, GitHub, Slack, Stripe, Kubernetes, HashiCorp Vault, Git, SSH credential stores
Published	2026-05-23T16:48:23
Discovery Source	Rss

Executive Summary

Attackers compromised four widely used Laravel Lang open-source packages by silently rewriting historical version tags on GitHub, redirecting up to 700 previously trusted package versions to malicious code without changing version numbers. Any developer environment that installed these packages, including environments using version pinning as a security control, may have received a credential-stealing payload targeting cloud keys, CI/CD secrets, SSH keys, browser-stored passwords, and cryptocurrency wallets. Organizations running Laravel-based applications should treat any environment that installed these packages before remediation as fully compromised and rotate all exposed credentials immediately.

Technical Analysis

The attack targeted laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, and laravel-lang/actions on Composer/Packagist. Attackers rewrote GitHub version tags, rather than publishing new versions, to point historical version references at malicious commits. This technique defeats version pinning (composer.lock) as a supply chain defense because the pinned version hash now resolves to attacker-controlled code. Up to 700 historical versions were affected. The payload is cross-platform and executes a credential-harvesting routine (T1195.001, Supply Chain Compromise; T1554, Compromise Client Software Binary) targeting: AWS, Stripe, Kubernetes, HashiCorp Vault, and GitHub credentials (T1552.001, T1528); SSH private keys (T1552.004);

Chrome, Brave, and Edge browser-stored credentials and cookies on Windows (T1555.003, T1539); and cryptocurrency wallet data. Obfuscation techniques were present (T1027). Exfiltration path not publicly confirmed but consistent with T1041 (Exfiltration Over C2 Channel). Relevant CWEs: CWE-494 (Download of Code Without Integrity Check), CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), CWE-345 (Insufficient Verification of Data Authenticity). No CVE assigned. Packagist has removed the malicious versions. Socket.dev identified and disclosed the compromise. Any install prior to Packagist remediation should be treated as a confirmed compromise.

Action Checklist

1. Containment: Immediately isolate any developer workstation, CI/CD pipeline, or build environment that ran 'composer install' or 'composer update' while laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, or laravel-lang/actions were listed as dependencies. Remove outbound internet access from affected systems pending forensic review. Revoke all cloud and service credentials accessible from those environments before completing investigation.
2. Detection: Audit composer.lock files across all repositories for entries referencing laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, or laravel-lang/actions at any version installed before Packagist remediation (confirmed removal date: 2026-05-23 per Socket.dev reporting). Query CI/CD pipeline logs and build artifact stores for installs of these packages. On Windows developer machines, review browser credential stores (Chrome, Brave, Edge) for unauthorized access events. Check SSH agent logs and ~/.ssh/known_hosts for anomalous entries. Review AWS CloudTrail, GitHub audit logs, Stripe Dashboard activity logs, and Kubernetes audit logs for API calls from unfamiliar IPs or service accounts (NIST AU-6, AU-12; CIS 8.2).
3. Eradication: Remove all four affected packages from dependency trees. Run 'composer remove laravel-lang/lang laravel-lang/http-statuses laravel-lang/attributes laravel-lang/actions' and delete composer.lock entries. Do not reinstall from Packagist until you have independently verified the current package integrity against the official Laravel-Lang GitHub repository. Rotate all secrets accessible from affected environments: AWS IAM keys and root credentials, GitHub personal access tokens and deploy keys, Slack tokens, Stripe API keys, Kubernetes service account tokens, HashiCorp Vault tokens, and all SSH private keys (NIST IA-5; D3-CRO, Credential Rotation). Purge browser-stored credentials and session cookies on affected Windows machines (D3-CH, Credential Hardening).
4. Recovery: After credential rotation, re-enable affected systems and monitor for 30 days using enhanced logging. Confirm that new composer installs resolve package hashes against expected values using 'composer validate' and verify integrity against the upstream GitHub repository commit history. Run 'composer show laravel-lang/lang --all' and cross-reference the returned commit hash against the official Laravel-Lang/lang repository's main branch history to confirm no tag rewriting remains. Re-run SAST/dependency scanning pipelines before promoting any build artifact produced in the affected window to production. Validate AWS, GitHub, Slack, Stripe, Kubernetes, and Vault access logs show no anomalous activity under the newly issued credentials (NIST SI-4; AU-6).
5. Post-Incident: This attack exposed a gap in assuming version pinning (composer.lock) provides supply chain integrity: it does not protect against tag rewriting on the source repository. Implement cryptographic verification of package integrity at install time using Composer's built-in hash verification combined with independent upstream verification. Adopt a policy of sourcing dependencies only from mirrored, internally verified registries for production pipelines (NIST CM-14, SR-4; CIS 2.1, 2.2; CWE-494 remediation). Evaluate adoption of SLSA (Supply-chain Levels for Software Artifacts) framework attestation

requirements for third-party Composer packages. Add laravel-lang package namespace to continuous dependency monitoring via Socket.dev or equivalent SCA tooling. Enable automatic dependency scanning via GitHub Dependabot, GitLab Dependency Scanning, or third-party SCA platforms (Socket.dev, Snyk) configured with security alert notifications. Tag-rewriting attacks do not change version numbers and will not trigger standard version-bump alerts.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, Legal, and external IR retainer immediately if AWS CloudTrail, GitHub audit logs, or Kubernetes audit logs show any API activity — under either old or newly rotated credentials — originating from IPs outside the organization's known ranges, indicating active credential use by the attacker and triggering breach notification assessment obligations under applicable data protection regulations.
Recovery Notes	After credential rotation and system re-enablement, apply enhanced logging at the API gateway and cloud control plane level for a minimum of 30 days, specifically monitoring for low-and-slow credential abuse patterns (e.g., infrequent AWS AssumeRole calls, GitHub API calls outside business hours) that indicate the attacker retained a credential not captured in the initial rotation sweep. Validate that all build artifacts produced during the exposure window — defined as any CI/CD build that ran composer install against the four affected packages prior to 2026-05-23 — are re-scanned with SAST and dependency analysis tools and are NOT promoted to production until cleared. Confirm integrity of the software supply chain recovery by verifying that composer.lock SHA entries for any reinstated laravel-lang packages match the cryptographic hashes of verified commits in the official upstream GitHub repository, not solely the Packagist registry.
Forensic Artifacts	Composer local cache ZIPs at ~/.composer/cache/files/laravel-lang/ (Linux/macOS) or %APPDATA%\Composer\cache\files\laravel-lang\ (Windows) — these files contain the actual malicious payload delivered under the rewritten tag and are the primary malware artifacts for reverse engineering and hash comparison against the legitimate package versions | Windows Security Event Log Event ID 4663 (An attempt was made to access an object) for read operations against Chrome Login Data at '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data', Brave at '%LOCALAPPDATA%\BraveSoftware\Brave-Browser\User Data\Default\Login Data', and Edge at '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Login Data' — access by any process other than the respective browser binary indicates credential harvesting by the stealer payload | AWS CloudTrail events filtered to EventNames: GetSecretValue, AssumeRole, CreateAccessKey, ListAccessKeys, and ConsoleLogin originating from developer workstation IPs or CI runner egress IPs during the composer install window — these represent the blast radius of the credential stealer's AWS exfiltration capability | File system entries created or modified within the timeframe of the malicious composer install — specifically any new executables, scripts, or data files in PHP temp directories (/tmp, %TEMP%), the project vendor/ directory, or user home directories that postdate the composer.lock timestamp and were not part of the expected vendor tree | Git object store anomalies in locally cloned laravel-lang repositories: run 'git fsck --full' and 'git log --walk-reflogs' to detect rewritten tag objects — the tag-rewriting attack leaves detached commit objects and orphaned refs in the git object database that differ from the original tag SHA recorded in composer.lock

Per-Action IR Details

Containment — Immediately isolate any developer workstation, CI/CD pipeline, or build environment that ran 'composer install' or 'composer update' while laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, or laravel-lang/actions were listed as dependencies. Remove outbound internet access from affected systems pending forensic review. Revoke all cloud and service credentials accessible from those environments before completing investigation.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-17 (Remote Access), NIST AC-3 (Access Enforcement), 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)

Compensating: On Windows developer machines, immediately disable NIC via: 'Disable-NetAdapter -Name * -Confirm:$false'. For CI/CD runners (GitHub Actions, GitLab CI, Jenkins), revoke the runner registration token and disable the runner job queue before isolating. Credential revocation priority order: AWS IAM access keys (aws iam delete-access-key), GitHub PATs (GitHub Settings > Developer Settings > Tokens), then SSH key deauthorization from ~/.ssh/authorized_keys on all servers that trusted the compromised machine. Document pre-isolation network state with 'netstat -ano > netstate_before_isolation.txt' for forensic baseline.

Evidence: Before isolating, capture: (1) Active network connections showing any C2 beacon or exfiltration channel established by the malicious laravel-lang payload — run 'netstat -ano' and correlate PIDs to processes via 'tasklist /FI "PID eq "'; (2) Running process list showing any PHP, composer, or shell process spawned during the malicious install window; (3) DNS cache ('ipconfig /displaydns') for any domains contacted by the stealer payload during or after composer execution; (4) Windows Prefetch files under C:\Windows\Prefetch\ for evidence of stealer binary execution (e.g., any unexpected .exe files with creation timestamps matching the composer install window); (5) Snapshot of environment variables ('Get-ChildItem Env:') to record which AWS_ACCESS_KEY_ID, GITHUB_TOKEN, or KUBECONFIG values were live in memory at time of compromise.

Detection — Audit composer.lock files across all repositories for entries referencing laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, or laravel-lang/actions at any version installed before Packagist remediation (confirmed removal date: 2026-05-23 per Socket.dev reporting). Query CI/CD pipeline logs and build artifact stores for installs of these packages. On Windows developer machines, review browser credential stores (Chrome, Brave, Edge) for unauthorized access events. Check SSH agent logs and ~/.ssh/known_hosts for anomalous entries. Review AWS CloudTrail, GitHub audit logs, Stripe Dashboard activity logs, and Kubernetes audit logs for API calls from unfamiliar IPs or service accounts (NIST AU-6, AU-12; CIS 8.2).

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: Enumerate all composer.lock files across repos with: 'find / -name composer.lock 2>/dev/null | xargs grep -l "laravel-lang"' (Linux/macOS) or 'Get-ChildItem -Recurse -Filter composer.lock | Select-String "laravel-lang"' (PowerShell). For CI/CD, grep build logs: 'grep -r "laravel-lang" ~/.jenkins/jobs/*/builds/*/log' or equivalent. For browser credential access detection on Windows without EDR, enable and query Windows Security Event Log for Event ID 4663 (Object Access — file read) against Chrome's Login Data file at '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data' and Edge's equivalent at '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Login Data'. For SSH anomalies, diff current ~/.ssh/known_hosts against a backup or Git-tracked version. Query AWS CloudTrail with: 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=ConsoleLogin' filtered to the install window timeframe.

Evidence: Specific artifacts for this tag-rewriting supply chain attack: (1) composer.lock SHA-256 hash of the laravel-lang packages — compare the 'dist.shasum' field in composer.lock against the legitimate package hash from the pre-compromise Packagist registry snapshot or Socket.dev report to confirm whether the rewritten tag was

installed; (2) Composer's local cache directory (~/.composer/cache/files/ on Linux, %APPDATA%\Composer\cache on Windows) — the cached package ZIP will contain the actual malicious payload files delivered under the rewritten tag; (3) Windows Security Event Log Event ID 4663 for read access to Chrome/Brave/Edge SQLite credential databases ('Login Data') by any process other than the browser itself; (4) AWS CloudTrail 'AssumeRole', 'GetSecretValue', or 'ListAccessKeys' events originating from developer workstation IPs or CI runner IPs during and after the composer install window; (5) GitHub audit log entries for 'oauth_access.create' or unexpected 'repo.clone' events using tokens that originated from the affected environment.

Eradication — Remove all four affected packages from dependency trees. Run 'composer remove laravel-lang/lang laravel-lang/http-statuses laravel-lang/attributes laravel-lang/actions' and delete composer.lock entries. Do not reinstall from Packagist until you have independently verified the current package integrity against the official Laravel-Lang GitHub repository. Rotate all secrets accessible from affected environments: AWS IAM keys and root credentials, GitHub personal access tokens and deploy keys, Slack tokens, Stripe API keys, Kubernetes service account tokens, HashiCorp Vault tokens, and all SSH private keys (NIST IA-5; D3-CRO — Credential Rotation). Purge browser-stored credentials and session cookies on affected Windows machines (D3-CH — Credential Hardening).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IA-5 (Authenticator Management), NIST CM-7 (Least Functionality), NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), CIS 5.2 (Use Unique Passwords), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: After running 'composer remove', purge Composer's local cache to ensure no malicious ZIP remains: 'composer clear-cache'. Verify removal: 'composer show | grep laravel-lang' should return empty. For credential rotation without an enterprise secrets manager: AWS IAM keys via 'aws iam create-access-key' then 'aws iam delete-access-key --access-key-id '; GitHub deploy keys via 'gh api /repos/{owner}/{repo}/keys' to list then DELETE each; SSH keys — generate new keypair with 'ssh-keygen -t ed25519', push new public key to all servers, then remove old public key from all authorized_keys files using: 'sed -i "/old_key_fingerprint/d" ~/.ssh/authorized_keys'. For HashiCorp Vault, revoke all tokens with known prefixes: 'vault token revoke -self' on affected machines then revoke root-derived tokens from Vault audit log. Browser credential purge on Windows: delete Chrome Login Data at '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data' and the associated 'Login Data-journal' file; repeat for Edge and Brave equivalent paths.

Evidence: Before eradication, capture: (1) Full contents of Composer cache for the four malicious packages (~/.composer/cache/files/laravel-lang/) — these ZIPs are the primary malware artifacts and must be preserved for reverse engineering before deletion; (2) File system timeline of any files dropped by the malicious package scripts (composer post-install-cmd or post-update-cmd hooks can write arbitrary files) — run 'find / -newer composer.lock -ls 2>/dev/null' scoped to the install timestamp to identify files created by the payload; (3) Contents of composer.json 'scripts' section in the malicious package (extracted from the cached ZIP) to document the exact payload execution mechanism used by the rewritten tag; (4) Windows Registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run for any persistence mechanism installed by the stealer payload; (5) Memory dump of any PHP or composer process still running, using ProcDump: 'procdump.exe -ma ' — the stealer may hold decrypted credentials in memory.

Recovery — After credential rotation, re-enable affected systems and monitor for 30 days using enhanced logging. Confirm that new composer installs resolve package hashes against expected values using 'composer validate' and verify integrity against the upstream GitHub repository commit history. Re-run SAST/dependency scanning pipelines before promoting any build artifact produced in the affected window to production. Validate AWS, GitHub, Slack, Stripe, Kubernetes, and Vault access logs show no anomalous activity under the newly issued credentials (NIST SI-4; AU-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 CM-3 (Configuration Change Control), NIST CP-10 (System Recovery and Reconstitution), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Hash verification for restored composer packages: run 'composer install --dry-run --no-plugins' and compare the resolved package hashes in composer.lock against the SHA-256 values published in the official laravel-lang GitHub repository's tagged release assets (not Packagist). Automate integrity checks in CI with a pre-build step: 'composer validate --strict && php -r "exit(shell_exec("composer audit") ? 0 : 1);"'. For 30-day monitoring without SIEM, create a daily cron job that diffs AWS CloudTrail events for new IAM key usage ('aws cloudtrail lookup-events --start-time $(date -d "yesterday" +%Y-%m-%dT%H:%M:%S)') and emails a digest. For Kubernetes, enable audit logging at RequestResponse level and review for new service account token issuance: 'kubectl get events --all-namespaces --field-selector reason=TokenRequest'. Use Sigma rule 'proc_creation_win_php_composer_install.yml' adapted to alert on any future laravel-lang package resolution.

Evidence: During recovery validation, document: (1) 'composer show --all laravel-lang/*' output confirming the packages are absent or show only post-remediation verified versions; (2) AWS CloudTrail event history for the newly rotated IAM access keys showing only expected API call patterns from known IPs — any API call from an unrecognized IP using the new key indicates the stealer exfiltrated the new credential during rotation, meaning the workstation is still compromised; (3) GitHub audit log 'git.push' or 'repo.create_fork' events under newly issued PATs — unauthorized use of new tokens within hours of issuance confirms persistent access; (4) Kubernetes audit log entries for the new service account tokens showing expected namespaces and verbs only; (5) Output of 'git log --all --oneline' on any locally cloned laravel-lang repository to verify no additional rewritten tags were introduced beyond the four identified packages.

Post-Incident — This attack exposed a gap in assuming version pinning (composer.lock) provides supply chain integrity: it does not protect against tag rewriting on the source repository. Implement cryptographic verification of package integrity at install time using Composer's built-in hash verification combined with independent upstream verification. Adopt a policy of sourcing dependencies only from mirrored, internally verified registries for production pipelines (NIST CM-14, SR-4; CIS 2.1, 2.2; CWE-494 remediation). Evaluate adoption of SLSA (Supply-chain Levels for Software Artifacts) framework attestation requirements for third-party Composer packages. Add laravel-lang package namespace to continuous dependency monitoring via Socket.dev or equivalent SCA tooling.

NIST Phase: Post Incident

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

Controls: NIST CM-14 (Signed Components), NIST SR-4 (Provenance), NIST SR-11 (Component Authenticity), NIST SI-2 (Flaw Remediation), NIST RA-3 (Risk Assessment), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Implement a local Composer mirror using Satis (open source): 'composer create-project composer/satis' — configure it to mirror only approved packages and serve as the sole Packagist endpoint in CI by setting 'COMPOSER_MIRROR_PATH_REPOS=1' and overriding the repository URL in composer.json. Enforce hash verification in all pipelines: add a CI gate that runs 'composer audit' and fails the build on any advisory match. Write a YARA rule targeting the specific stealer behavior (reading SQLite browser credential databases from known Windows paths) for scanning developer workstations post-incident. Adopt a GitHub Actions workflow-level pin to commit SHAs instead of tags (e.g., 'uses: actions/checkout@' not '@v3') to prevent analogous tag-rewriting in CI dependencies. Add a pre-commit hook that rejects composer.lock changes introducing new laravel-lang/* entries without a mandatory peer review approval: 'git config core.hooksPath .githooks' with a shell script checking 'git diff --cached composer.lock | grep laravel-lang'.

Evidence: Post-incident documentation to retain: (1) The malicious package ZIPs preserved from Composer cache — these are the primary forensic artifacts proving the attack and should be submitted to the Socket.dev or GitHub Security Advisory Database for community protection; (2) A complete timeline reconstructed from AWS CloudTrail, GitHub audit logs, and Composer cache timestamps showing the exact install-to-exfiltration window for the credential stealer; (3) Diff of the rewritten git tags versus the legitimate tag objects — obtain via 'git ls-remote

https://github.com/Laravel-Lang/lang refs/tags/*' compared against a pre-compromise snapshot to document the exact tag manipulation technique used; (4) Lessons-learned report documenting which composer.lock-pinned versions were affected, to update internal developer guidance that version pinning is insufficient without cryptographic hash validation; (5) Threat intelligence report for internal distribution mapping this attack to MITRE ATT&CK T1195.001 (Compromise Software Dependencies and Development Tools) and T1552.001 (Credentials In Files) for future detection rule development.

Detection Guidance

Primary detection signal: presence of laravel-lang/lang, laravel-lang/http-statuses, laravel-lang/attributes, or laravel-lang/actions in any composer.lock file across your codebase, CI/CD configuration, or build artifacts. Search all repositories and artifact stores. Secondary signals, behavioral indicators consistent with credential theft payload execution: (1) Unexpected outbound connections from developer workstations or build agents to unknown IPs on ports 443/80 during or after a composer install job (T1041). (2) Anomalous API calls in AWS CloudTrail: ListBuckets, GetSecretValue, AssumeRole from unfamiliar source IPs or at unusual hours. (3) GitHub audit log entries showing personal access token use from IPs not matching developer locations (T1528). (4) SSH authentication attempts from unknown source IPs using keys stored on affected developer machines (T1552.004). (5) Browser credential store access events on Windows, review Windows Event Log for DPAPI-related events (Event ID 4693, 4694) on developer machines that ran Composer during the exposure window (T1555.003). (6) Kubernetes audit logs showing service account token use from pod names or source IPs inconsistent with normal workload patterns. IOCs: No confirmed hashes, domains, or IPs have been publicly released as of the reporting date (2026-05-23). Monitor Socket.dev advisory at https://socket.dev/blog/laravel-lang-compromise for updated IOC releases. Treat absence of published IOCs as expected for an early-stage disclosure; absence of a match does not rule out compromise. For ongoing monitoring of tag-rewrite attacks in Composer dependencies, enable automatic dependency scanning via GitHub Dependabot, GitLab Dependency Scanning, or third-party SCA platforms (Socket.dev, Snyk) configured with security alert notifications. Tag-rewriting attacks do not change version numbers and will not trigger standard version-bump alerts. Framework references: NIST AU-6 (Audit Record Review), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), MITRE T1195.001, T1554, T1552.001.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://socket.dev/blog/laravel-lang-compromise`	Socket.dev primary advisory — check for updated IOC releases including malicious commit hashes, C2 infrastructure, and additional affected package versions	HIGH
URL	`https://packagist.org/packages/laravel-lang/`	Packagist package listing — verify current package integrity and confirm malicious versions have been removed before reinstalling	HIGH

Framework Mappings

MITRE-ATTACK

- **T1539** — Steal Web Session Cookie
- **T1528** — Steal Application Access Token
- **T1027** — Obfuscated Files or Information
- **T1059.004** — Unix Shell
- **T1552.004** — Private Keys
- **T1555.003** — Credentials from Web Browsers
- **T1087.001** — Local Account
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1554** — Compromise Host Software Binary
- **T1552.001** — Credentials In Files
- **T1041** — Exfiltration Over C2 Channel
- **T1496** — Resource Hijacking

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **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
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **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

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
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1539	Steal Web Session Cookie	Credential-Access
T1528	Steal Application Access Token	Credential-Access
T1027	Obfuscated Files or Information	Defense-Evasion
T1059.004	Unix Shell	Execution
T1552.004	Private Keys	Credential-Access
T1555.003	Credentials from Web Browsers	Credential-Access
T1087.001	Local Account	Discovery
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1554	Compromise Host Software Binary	Persistence
T1552.001	Credentials In Files	Credential-Access
T1041	Exfiltration Over C2 Channel	Exfiltration
T1496	Resource Hijacking	Impact

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/laravel-lang-package...	T3
laravel-lang/common - Packagist.org	https://packagist.org/packages/laravel-lang/common	T3
Laravel Lang Compromised with RCE Backdoor Across 700+ Versions	https://socket.dev/blog/laravel-lang-compromise	T3
GitHub - Laravel-Lang/lang: List of 128 languages for Laravel ...	https://github.com/Laravel-Lang/lang	T3

Source	URL	Tier
Packages from laravel-lang - Packagist	https://packagist.org/packages/laravel-lang/	**T3**

Generated 2026-05-24 06:19 UTC by TJS Security Command Center