INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:33 UTC

Laravel Livewire Remote Code Execution via Code Injection (CVE-2025-54068)

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8 | **CISA KEV**

SCC Item ID	SCC-CVE-2026-0022
Type	CVE Vulnerability
CVE ID	CVE-2025-54068
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.1597 (95th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability (due: 2026-04-03)
Affected Products	Laravel Livewire < 3.6.4
Published	2026-03-20

Executive Summary

A critical remote code execution vulnerability (CVE-2025-54068, CVSS 9.8) affects Laravel Livewire versions prior to 3.6.4, allowing unauthenticated attackers to execute arbitrary commands on web servers running affected configurations. CISA has added this vulnerability to the Known Exploited Vulnerabilities catalog, confirming active exploitation in the wild. Organizations running Laravel-based web applications face potential full server compromise, data exfiltration, and supply chain risk if patching is not completed immediately.

Technical Analysis

CVE-2025-54068 is a code injection vulnerability (CWE-94) rooted in insecure deserialization of component state (CWE-502) within Laravel Livewire 3.x prior to 3.6.4. The flaw exists in how Livewire unmarshals serialized component state transmitted between client and server. An unauthenticated attacker can craft malicious serialized payloads and submit them to Livewire component endpoints, triggering arbitrary PHP code execution on the server. No authentication or user interaction is required in affected configurations. Exploitation maps to MITRE ATT&CK T1190 (Exploit Public-Facing Application) for initial access and T1059 (Command and Scripting Interpreter) for execution. EPSS score of 0.15973 (94.7th percentile) indicates high relative exploitation probability compared to all known CVEs. The vulnerability was discovered and disclosed by Synacktiv. Patch is available: Livewire 3.6.4. No patch exists for Livewire 2.x; users on that branch should assess exposure separately. CISA KEV remediation due date is 2026-04-03.

Action Checklist

1. Step 1 (Immediate): Upgrade all Laravel Livewire installations to version 3.6.4 or later. If immediate patching is not possible, restrict access to Livewire component endpoints at the WAF or network layer as a temporary mitigation.
2. Step 2 (Detection): Review web server and application logs for anomalous POST requests targeting Livewire component endpoints (typically paths containing /livewire/message or /livewire/update). Look for oversized or malformed serialized payloads and unexpected command execution artifacts in application logs.
3. Step 3 (Assessment): Inventory all internal and externally facing Laravel applications to identify Livewire 3.x deployments below version 3.6.4. Include third-party and vendor-managed applications in scope.
4. Step 4 (Communication): Notify application owners, development teams, and relevant stakeholders of the CISA KEV status and the 2026-04-03 remediation deadline. Escalate to CISO if any externally exposed instances are confirmed unpatched.
5. Step 5 (Long-term): Review PHP deserialization handling across all Laravel applications. Evaluate WAF rules for PHP object injection and deserialization attack patterns. Incorporate Livewire version pinning and automated dependency scanning into CI/CD pipelines to prevent future exposure from component library updates.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and initiate emergency incident response if any externally exposed unpatched Livewire instances are confirmed, or if any post-exploitation indicators (command execution in logs, unauthorized file writes, credential access) are detected during Step 2 log analysis.
Recovery Notes	After patching is complete, conduct 72-hour post-remediation log review comparing pre- and post-patch application behavior to detect any residual backdoors or persistence mechanisms. Validate that all WAF restrictions have been removed and patch effectiveness is confirmed (version checks and functional testing of Livewire components). Schedule architecture review with development teams to evaluate long-term PHP hardening and dependency management strategy within 30 days.
Forensic Artifacts	/var/log/apache2/access.log and /var/log/apache2/error.log (HTTP request/response metadata and PHP errors) | /var/log/php-fpm.log and /var/log/laravel.log (application exceptions, serialization warnings, command execution traces) | /var/log/auth.log and /var/log/syslog (command execution, privilege escalation attempts, system activity) | composer.lock and composer.json (package version baseline for forensic timeline and supply chain verification) | tcpdump capture files or WAF logs (HTTP request bodies, payload reconstruction, exploit pattern detection)

Per-Action IR Details

Step 1 (Immediate): Upgrade all Laravel Livewire installations to version 3.6.4 or later. If immediate patching is not possible, restrict access to Livewire component endpoints at the WAF or network layer as a temporary mitigation.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.2.3 (Containment Strategies)

Controls: NIST SI-2 (Flaw Remediation), NIST AC-3 (Access Enforcement), CIS 6.2 (Patch Management), CIS 4.1 (Establish and Maintain a Secure Configuration Management Process)

Compensating: If WAF unavailable, use nginx/Apache rewrite rules to block POST requests to /livewire/message and /livewire/update: `nginx: location ~ /livewire/(message|update) { return 403; }`. Alternatively, use iptables to rate-limit or drop traffic to affected endpoints: `iptables -A INPUT -p tcp --dport 80 -m string --string 'livewire' --algo bm -j DROP`. Verify with `curl -X POST http://target/livewire/message -v` — expect 403 or connection timeout.

Evidence: Capture current Composer lock file (`composer.lock`) to document pre-patch version baseline. Enable query logging on web server (`Apache: LogLevel debug`, `nginx: access_log /var/log/nginx/access.log with $request_body`) to capture POST payloads before and after patching. Preserve application error logs and PHP-FPM slow logs for 72 hours pre-patch as baseline for post-patch comparison.

Step 2 (Detection): Review web server and application logs for anomalous POST requests targeting Livewire component endpoints (typically paths containing /livewire/message or /livewire/update). Look for oversized or malformed serialized payloads and unexpected command execution artifacts in application logs.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.2 (Detection and Analysis)

Controls: NIST AU-2 (Audit Events), NIST AU-6 (Audit Review, Analysis, and Reporting), NIST SI-4 (Information System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 8.5 (Configure Log Alert Aggregation)

Compensating: Use `grep` and `awk` to parse web server access logs for Livewire endpoints: `grep -E 'livewire/(message|update)' /var/log/apache2/access.log | awk '{print $1, $4, $7, $9, $10}' > livewire_requests.txt`. Filter for POST method and large Content-Length: `grep 'POST.*livewire' /var/log/apache2/access.log | awk '$10 > 50000 {print $0}' > oversized_posts.txt`. Check PHP error log for serialization warnings: `grep -i 'unserialize\|object injection\|unexpected token' /var/log/php-fpm.log`. For command execution, search application logs for shell metacharacters in request parameters: `grep -E '(;|`|\$\(|&&|\|\|)' /var/log/laravel.log | grep livewire`.

Evidence: Preserve full HTTP request/response bodies for all POST requests to /livewire/message and /livewire/update from the past 30 days (or since deployment, whichever is shorter). Extract into separate file: `tcpdump -i any 'tcp port 80 or 443' -w livewire_traffic.pcap`. Capture PHP-FPM slow log entries with request URI, duration, and stack traces. Log application-level exceptions and stack traces from Laravel logs. Document system command execution timestamps from auth logs and syslog: `grep -E 'sudo|COMMAND=' /var/log/auth.log | grep -v 'sudo: session opened'`.

Step 3 (Assessment): Inventory all internal and externally facing Laravel applications to identify Livewire 3.x deployments below version 3.6.4. Include third-party and vendor-managed applications in scope.

NIST Phase: Preparation

Reference: NIST 800-61r3 §3.1 (Preparation)

Controls: NIST CM-8 (Information System Component Inventory), NIST SA-4 (Acquisition Process), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain Software Inventory)

Compensating: Query application package repositories manually: For each web server, execute `find / -name 'composer.json' 2>/dev/null -exec grep -l 'livewire' {} \;` to locate all Laravel+Livewire installations. For each match, run `composer show livewire` or inspect `/vendor/livewire/livewire/composer.json` to extract version. Create inventory spreadsheet with columns: hostname, application name, install path, current Livewire version, patch status, externally exposed (yes/no), owner contact. Use network scanning with nmap service detection: `nmap -p 80,443 --script http-robots.txt --script http-title -oX scan_results.xml ` to identify web applications, then correlate with asset inventory.

Evidence: Preserve each `composer.lock` and `composer.json` file from all identified installations as point-in-time baseline. Capture version output from each installation: `php artisan tinker <<< 'echo Livewire\Livewire::VERSION;'` or similar. Document network exposure: `netstat -tlnp | grep LISTEN` and compare listening ports to firewall rules and asset inventory. Archive all results with timestamps for audit trail.

Step 4 (Communication): Notify application owners, development teams, and relevant stakeholders of the CISA KEV status and the 2026-04-03 remediation deadline. Escalate to CISO if any externally exposed instances are confirmed unpatched.

NIST Phase: Preparation

Reference: NIST 800-61r3 §3.4.2 (Post-Incident Activity: Lessons Learned) and NIST 800-53 IR-4 (Incident Handling)

Controls: NIST IR-4 (Incident Handling), NIST CA-7 (Continuous Monitoring), CIS 5.4 (Establish and Maintain an Incident Response Process)

Compensating: Create formal incident notification using email template with: CVE-2025-54068 details, CVSS 9.8 rating, link to CISA KEV entry (https://www.cisa.gov/known-exploited-vulnerabilities), remediation deadline (2026-04-03), current inventory status for their application, required action (upgrade to 3.6.4), and escalation contact (CISO email/phone). Use mail merge or distribution list to ensure consistent messaging. Document all recipient acknowledgments with timestamp. For externally exposed instances, generate separate urgent notification to CISO with risk summary: 'X unpatched externally exposed Livewire instances detected; immediate WAF restriction recommended pending patch deployment.'

Evidence: Preserve all notification emails, distribution lists, and delivery receipts. Maintain a communication log with recipient name, role, notification timestamp, and acknowledgment status. For escalated cases, document the decision (e.g., email to CISO with response) and any temporary mitigations implemented as a result.

Step 5 (Long-term): Review PHP deserialization handling across all Laravel applications. Evaluate WAF rules for PHP object injection and deserialization attack patterns. Incorporate Livewire version pinning and automated dependency scanning into CI/CD pipelines to prevent future exposure from component library updates.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.4 (Post-Incident Activity) and NIST 800-53 SI-2 (Flaw Remediation)

Controls: NIST SI-2 (Flaw Remediation), NIST SA-11 (Software Development, Testing, and Validation), NIST SC-7 (Boundary Protection), CIS 2.4 (Address Unauthorized Software), CIS 6.2 (Patch Management), CIS 13.3 (Deploy and Maintain System Monitoring Tools)

Compensating: For PHP deserialization review, add input validation to prevent untrusted serialized object injection: implement strict allowlist of permitted classes in unserialize() calls, or replace with `json_decode()` where possible. Create WAF rules to block requests with PHP serialization markers ('O:' or 'C:' object prefix, common in PHP object notation). Example ModSecurity rule: `SecRule REQUEST_BODY '@rx (?:O|C):\d+:\"' 'id:1001,deny,status:403,msg:PHP_Serialization_Attempt'`. For CI/CD, add `composer audit` command to pre-commit hooks and pipeline stages to check for known vulnerabilities before deployment. Implement version pinning: modify `composer.json` to require exact version `'livewire/livewire': '3.6.4'` (not `^3.0`), then use `composer update --lock` to prevent accidental minor version upgrades. Set up automated weekly dependency scanning with `Dependabot` or `Snyk` and configure alerts to security team.

Evidence: Document baseline WAF ruleset and PHP configuration (`php.ini` settings for `disable_functions`, `open_basedir`, `suhosin` if applicable). Capture pre-existing deserialization validation logic from application codebase with git commit hashes. Archive updated `composer.json` with pinned versions and timestamps. Log all WAF rule deployments and test results (false positive checks). Preserve CI/CD pipeline configuration files showing new security gates and their execution logs.

Detection Guidance

Focus detection on Livewire's AJAX component update endpoints. In Apache or Nginx access logs, search for POST requests to paths matching '/livewire/message' or '/livewire/update' originating from unexpected or unauthenticated sources. Look for payloads that are base64-encoded or contain PHP serialization markers ('O:' prefix in request bodies), which indicate potential object injection attempts. In application-level logs, watch for unexpected process spawning from the web server user (e.g., 'www-data' or 'apache' spawning shell

processes). On Linux hosts, audit logs (auditd) can surface execve calls initiated from PHP-FPM or Apache worker processes. If a WAF is in place, enable PHP deserialization and code injection rule sets and alert on matches against Livewire endpoint patterns. No specific IOCs (IPs, domains, hashes) have been publicly attributed to active exploitation campaigns at time of this report; behavioral detection is the primary signal.

Framework Mappings

MITRE-ATTACK

- **T1059** — Command and Scripting Interpreter
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-10** — Information Input Validation
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A03:2021** — Injection

CIS-V8

- **16.10**

ISO-27001-2022

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

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1059	Command and Scripting Interpreter	Execution
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
cisa_kev	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1
CVE-2025-54068 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2025-54068	T1
CVE-2025-54068: Laravel Livewire RCE Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2025-54068/	T3
Livewire: remote command execution through unmarshaling	https://synacktiv.com/advisories/livewire-remote-command-execution-...	T3
Livewire 3.x < 3.6.4 Remote Code Execution	https://www.tenable.com/plugins/was/115113	T3

Generated 2026-03-29 18:33 UTC by TJS Security Command Center