INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-21 06:59 UTC

Highly Critical Drupal Core RCE Vulnerability Affecting PostgreSQL-Backed Sites

CVE VULNERABILITY | CRITICAL | CVSS 9.0

SCC Item ID	SCC-CVE-2026-0205
Type	CVE Vulnerability
Severity	CRITICAL
CVSS Base Score	9.0
Affected Products	Drupal Core (PostgreSQL configurations), specific versions unconfirmed from available sources
Published	6 hours ago
Discovery Source	Serper

Executive Summary

Drupal has issued a 'highly critical' security advisory for a remote code execution vulnerability in Drupal Core affecting sites running PostgreSQL as the database backend. Unauthenticated or low-privilege attackers can exploit this flaw over the network, potentially gaining full control of affected web servers. Organizations running Drupal on PostgreSQL should treat this as a priority patching event; a CVE ID and confirmed version ranges are pending direct verification from drupal.org/security. Monitor drupal.org/security for the official advisory publication, which will specify affected versions, patch availability, and assigned CVE.

Technical Analysis

Drupal has published a 'highly critical' advisory for a remote code execution vulnerability in Drupal Core (CWE-94: Improper Control of Code Generation) affecting PostgreSQL-backed deployments. The attack vector aligns with MITRE ATT&CK T1190 (Exploit Public-Facing Application). A CVSS base score of 9.0 is associated with this item; the attack appears exploitable by unauthenticated or low-privilege network-adjacent attackers. A CVE ID has not been confirmed from available source excerpts; the drupal.org/security advisory page is the authoritative source for affected version ranges, patch availability, and the assigned CVE. The primary reporting URL (thehackernews.com/2026/05/) could not be actively verified and should not be used as a source of record. Confidence in technical specifics is LOW pending direct advisory verification from drupal.org/security or NVD. Secondary sources (The Hacker News, Reddit) are T3-tier references only. Note: Prior Drupal RCE vulnerabilities (e.g., CVE-2019-6340) are distinct from this advisory and historical source material should not be used to characterize the current vulnerability.

Action Checklist

1. Step 1: Containment. Immediately identify all Drupal instances in your environment using PostgreSQL as the database backend (query your CMDB or asset inventory per CIS 1.1). For internet-facing instances, consider applying WAF rules to REST API and JSON:API endpoints per OWASP Drupal hardening guidance, or restrict access by IP, or take affected public-facing sites offline until the official patch is available and applied (NIST AC-4: Information Flow Enforcement).
2. Step 2: Detection. Query web server and application logs for unexpected or anomalous requests to Drupal REST API or JSON:API endpoints, especially from unauthenticated sources or low-privilege accounts. Look for unexpected process spawning from the web server process (e.g., PHP spawning shell processes). Enable or review audit logging per NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs). Check for unusual file creation, modification of core files, or new PHP files in the Drupal docroot. Confirm no CVE ID or IOCs are available yet from drupal.org; monitor the official advisory for updated indicators as it matures.
3. Step 3: Eradication. Apply the official Drupal security update as soon as it is published at drupal.org/security. Do not rely on third-party patch announcements; pull the update directly from Drupal's official advisory, which will specify the exact affected versions and the patched release. Verify file integrity of Drupal core after patching using Drupal's built-in file integrity checker or a hash comparison against the official release. Review and remove any unauthorized files discovered during detection.
4. Step 4: Recovery. After patching, validate the Drupal application returns expected responses and that no unauthorized accounts, backdoors, or scheduled tasks were created during any exploitation window. Rotate database credentials and any API keys accessible to the Drupal application. Resume normal operations only after integrity checks pass and logging confirms no ongoing anomalous activity (NIST AU-6: Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident. Evaluate whether a Web Application Firewall was in place and tuned for Drupal-specific attack patterns; if not, prioritize deployment. Review account privilege configurations on Drupal to enforce least privilege (NIST AC-6). Assess whether unauthenticated or low-privilege API access is necessary for your use case and restrict accordingly (NIST AC-3: Access Enforcement). Document findings and update your vulnerability management process to include monitoring of drupal.org/security advisories on a recurring basis (CIS 7.1: Establish and Maintain a Vulnerability Management Process).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and activate breach notification procedures immediately if forensic evidence shows successful exploitation — specifically: webshell files found in docroot, unauthorized Drupal admin accounts created, or PostgreSQL query logs showing data exfiltration (e.g., COPY TO or large SELECT on user/content tables) — given a CVSS 9.0 unauthenticated RCE against a publicly reachable web application storing PII or sensitive content.

Recovery Notes	After patching and credential rotation, maintain elevated logging verbosity on the Drupal application (PHP error logging, PostgreSQL log_min_duration_statement=0, and web server access logging) for a minimum of 30 days, reviewing daily for recurrence of the anomalous JSON:API or REST POST patterns identified during detection. Validate the integrity of all Drupal-managed content and user data in PostgreSQL by comparing row counts and last-modified timestamps against pre-incident backups to detect any unauthorized modification or exfiltration. Do not reduce monitoring posture until the official CVE ID is published and threat intelligence sources confirm no active exploitation campaigns are ongoing against this specific vulnerability.
Forensic Artifacts	Web server access logs (Apache /var/log/apache2/access.log or nginx /var/log/nginx/access.log): filter for POST requests to /jsonapi/, /rest/session/token, and /api/ paths — specifically HTTP 200/201 responses from sessions lacking valid Drupal session cookies, which indicates unauthenticated exploitation of the Drupal Core RCE. | PHP-FPM process tree artifacts: output of 'ps auxf' or auditd EXECVE records showing php-fpm or php worker processes spawning unexpected child processes (bash, sh, curl, wget, python) — the signature process chain an RCE exploit against a Drupal PHP application would produce. | PostgreSQL server logs (/var/log/postgresql/postgresql-*.log with log_statements='all'): look specifically for COPY TO/FROM PROGRAM commands, lo_import/lo_export large object operations, or CREATE FUNCTION calls issued under the Drupal application DB role, which represent the PostgreSQL-specific code execution primitives this vulnerability class likely leverages. | Drupal docroot filesystem timeline: output of 'find /var/www/html -name "*.php" -newer -ls' — new or modified PHP files outside of the official Drupal release manifest in sites/default/files/, modules/, or themes/ directories indicate webshell implantation following successful RCE exploitation. | Drupal database users_field_data and user__roles tables (PostgreSQL): records with created timestamps falling within the exploitation window and role assignments to 'administrator' identify persistence mechanisms — attacker-created admin accounts are a standard post-exploitation persistence step following Drupal RCE.

Per-Action IR Details

Step 1: Containment — Immediately identify all Drupal instances in your environment using PostgreSQL as the database backend (query your CMDB or asset inventory per CIS 1.1). For internet-facing instances, apply WAF rules to block anomalous POST requests to Drupal API endpoints while patch assessment proceeds. If no WAF is in place, consider taking affected public-facing sites offline or restricting access by IP until patching is complete (NIST AC-4: Information Flow Enforcement).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST IR-4 (Incident Handling), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Without a WAF, use nginx or Apache access controls to immediately restrict Drupal's /jsonapi/, /api/, and /rest/ URI paths to known-good IP ranges: nginx — 'location ~* ^/(jsonapi|api|rest)/ { allow ; deny all; }'; Apache — add 'Require ip ' inside a block for those paths. For asset discovery without a CMDB, run: 'grep -r "pgsql\|pdo_pgsql\|postgres" /var/www/*/sites/*/settings.php' across your web root to enumerate all PostgreSQL-backed Drupal instances in under five minutes.

Evidence: Before restricting access, capture a full snapshot of current active HTTP connections to the Drupal server: 'ss -tnp | grep :80\|:443' and 'netstat -antp | grep php-fpm' to document any live sessions that may represent active exploitation. Dump the PostgreSQL pg_stat_activity view ('SELECT pid, usename, application_name, client_addr, query, state FROM pg_stat_activity;') to baseline current database connections before containment alters the session state.

Step 2: Detection — Query web server and application logs for unexpected or anomalous requests to Drupal REST API or JSON:API endpoints, especially from unauthenticated sources or low-privilege accounts. Look for unexpected process spawning from the web server process (e.g., PHP spawning shell processes). Enable or review audit logging per NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs). Check for unusual file creation, modification of core files, or new PHP files in the Drupal docroot (D3-SFA: System File Analysis). Confirm no CVE ID or IOCs are available yet from drupal.org — monitor the advisory for updated indicators.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Run the following to detect PHP webshells or new files dropped in the Drupal docroot since the advisory date: 'find /var/www/html -name "*.php" -newer /var/www/html/index.php -ls'. To detect PHP spawning child shells (indicative of RCE), enable Sysmon on Linux via sysmon-ebpf or use auditd: add rule 'auditctl -a always,exit -F arch=b64 -S execve -F ppid=$(pgrep -d, php-fpm) -k drupal_rce' to catch any process spawned by php-fpm. Check PostgreSQL logs at /var/log/postgresql/postgresql-*.log for unexpected COPY TO/FROM PROGRAM statements or large_object operations that may indicate DB-layer code execution.

Evidence: Preserve the following BEFORE any remediation: (1) Full Apache/nginx access logs (/var/log/apache2/access.log or /var/log/nginx/access.log) — grep for POST requests to /jsonapi/, /rest/, /api/; filter by HTTP 200/201/500 responses from unauthenticated sessions (no session cookie or X-CSRF-Token). (2) PHP-FPM slow log and error log (/var/log/php-fpm/*.log) for stack traces that may reveal the exploited code path. (3) PostgreSQL server log with log_min_duration_statement=0 output to capture any anomalous queries issued by the Drupal DB user. (4) Output of 'find /var/www -name "*.php" -mtime -7' timestamped at collection time to document recent file modifications.

Step 3: Eradication — Apply the official Drupal security update as soon as it is published at drupal.org/security. Do not rely on third-party patch announcements; pull the update directly from Drupal's official advisory, which will specify the exact affected versions and the patched release. Verify file integrity of Drupal core after patching using Drupal's built-in file integrity checker or a hash comparison against the official release (D3-FMBV: File Magic Byte Verification). Review and remove any unauthorized files discovered during detection.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: After applying the Drupal update via 'drush pm:update drupal/core --no-dev' or Composer ('composer update drupal/core-recommended'), verify core file integrity without a commercial tool: download the official Drupal release tarball matching your version from drupal.org, then run 'diff -rq --exclude="*.txt" /var/www/html/core/ drupal-/core/' to identify any files that differ from the canonical release. For any PHP files found during detection that are not in the official release manifest, collect them as forensic artifacts ('cp -p /evidence/') before deleting.

Evidence: Before patching, image or snapshot the affected web server's docroot ('tar czf /evidence/drupal_docroot_prepatching_$(date +%Y%m%d%H%M).tar.gz /var/www/html/') and capture the PostgreSQL drupal database schema ('pg_dump -s -U drupaluser drupal > /evidence/drupal_schema_prepatching.sql') to preserve pre-patch state for forensic comparison. Document the exact Drupal version in place at time of patching: 'drush status | grep version'.

Step 4: Recovery — After patching, validate the Drupal application returns expected responses and that no unauthorized accounts, backdoors, or scheduled tasks were created during any exploitation window (D3-LAM: Local Account Monitoring). Rotate database credentials and any API keys accessible to the Drupal application (D3-CRO: Credential Rotation). Resume normal operations only after integrity checks pass and logging confirms no ongoing anomalous activity (NIST AU-6: Audit Record Review, Analysis, and Reporting).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST IA-5 (Authenticator Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 5.3 (Disable Dormant Accounts)

Compensating: Query the Drupal database directly to enumerate all user accounts created or modified during the exploitation window — using psql: 'SELECT uid, name, mail, created, access FROM users_field_data WHERE created > EXTRACT(EPOCH FROM NOW() - INTERVAL "7 days") ORDER BY created DESC;' — and cross-reference against your known account inventory. Check OS-level cron for backdoor persistence: 'crontab -l -u www-data; ls -la /etc/cron.*; cat /var/spool/cron/crontabs/*'. Rotate the Drupal DB password in settings.php and revoke the old PostgreSQL role credential: 'ALTER ROLE drupaluser PASSWORD "";'.

Evidence: Before rotating credentials, dump the full Drupal users table and any role/permission assignments ('SELECT u.uid, u.name, r.roles_target_id FROM users_field_data u JOIN user__roles r ON u.uid = r.entity_id;') to document the account state at time of recovery. Review OS authentication logs (/var/log/auth.log or /var/log/secure) for any SSH logins or sudo usage by the www-data account during the suspected exploitation window, which would indicate privilege escalation beyond the web process.

Step 5: Post-Incident — Evaluate whether a Web Application Firewall was in place and tuned for Drupal-specific attack patterns; if not, prioritize deployment. Review account privilege configurations on Drupal to enforce least privilege (NIST AC-6). Assess whether unauthenticated or low-privilege API access is necessary for your use case and restrict accordingly (NIST AC-3: Access Enforcement, D3-UAP: User Account Permissions). Document findings and update your vulnerability management process to include monitoring of drupal.org/security advisories on a recurring basis (CIS 7.1: Establish and Maintain a Vulnerability Management Process).

NIST Phase: Post Incident

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

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

Compensating: Without a commercial WAF, deploy ModSecurity with the OWASP Core Rule Set (CRS) free — enable the Drupal-specific exclusion rules from the CRS project and add custom rules targeting Drupal JSON:API and REST endpoint abuse patterns. Subscribe to the drupal.org/security RSS feed (https://www.drupal.org/security/rss.xml) and automate alerting via a cron job that diffs the feed daily. For Drupal API hardening without enterprise tooling, disable JSON:API entirely if not required: set 'jsonapi.settings' — 'read_only: true' in services.yml, or uninstall the JSON:API module if unauthenticated API access has no operational justification.

Evidence: Compile the lessons-learned artifact package: final diff of Drupal core files against official release (from eradication step), timeline of all anomalous POST requests to API endpoints extracted from preserved access logs, list of any accounts created or modified during exploitation window, and PostgreSQL audit log excerpt covering the incident window. This package supports both internal review and any regulatory notification obligations if PII stored in the Drupal database was accessible during the exploitation window.

Detection Guidance

Primary detection focus: web server access logs and application logs for Drupal REST API and JSON:API endpoints. Look for POST or PATCH requests from unauthenticated sessions or low-privilege accounts to paths such as /jsonapi/, /node/, or /user/ that result in unexpected 200 or 201 responses. Look for web server child processes (PHP-FPM, Apache mod_php) spawning unexpected system commands (e.g., bash, sh, cmd.exe). Monitor for new or modified PHP files in the Drupal docroot, especially in writable directories (NIST AU-2). Enable system call auditing (auditd on Linux) to flag unexpected process executions from the web server user.

No confirmed IOCs (IPs, file hashes, domains) are available from verified sources at this time; monitor drupal.org/security and CISA advisories for updated indicators as the official advisory is published and the CVE is assigned. CIS 8.2 compliance (audit log collection across enterprise assets) is a prerequisite for effective detection here.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.drupal.org/security`	Official Drupal security advisory page — monitor for CVE ID, affected versions, and patch release specific to this vulnerability	HIGH

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A03:2021** — Injection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures

ISO-27001-2022

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
	https://thehackernews.com/2026/05/highly-critical-drupal-core-flaw....	T3
Highly Critical Drupal Core Flaw Exposes PostgreSQL Sites to RCE ...	https://www.reddit.com/r/hackerworkspace/comments/1tjb8tc/highly_cr...	T3
'Highly critical' bug exposes unpatched Drupal sites to attacks	https://www.welivesecurity.com/2019/02/27/highly-critical-bug-expos...	T3
Security advisories | Drupal.org	https://www.drupal.org/security	T3
Drupal Vulnerability Can Be Exploited for RCE Attacks - Trend Micro	https://www.trendmicro.com/en_us/research/19/b/drupal-vulnerability...	T3

Generated 2026-05-21 06:59 UTC by TJS Security Command Center