INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-29 19:06 UTC

CVE-2026-45247: Mirasvit Full Page Cache Warmer PHP Object Injection Enables Unauthenticated RCE

CVE VULNERABILITY | CRITICAL | CVSS 9.8 | CISA KEV

SCC Item ID	SCC-CVE-2026-0239
Type	CVE Vulnerability
CVE ID	CVE-2026-45247
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0010 (28th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	Mirasvit Full Page Cache Warmer for Magento 2, versions before 1.11.12
Published	2026-05-29T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A critical unauthenticated remote code execution vulnerability in the Mirasvit Full Page Cache Warmer extension for Magento 2 allows attackers to take full control of affected e-commerce servers without any login credentials. Any internet-facing Magento 2 storefront running Cache Warmer versions before 1.11.12 is at immediate risk of server compromise, data theft, payment skimming, and complete operational shutdown. CISA has added this to its Known Exploited Vulnerabilities catalog, confirming active exploitation in the wild.

Technical Analysis

CVE-2026-45247 (CVSS 9.8, CWE-502) is a PHP object injection vulnerability in Mirasvit Full Page Cache Warmer for Magento 2, affecting all versions before 1.11.12. The extension passes the CacheWarmer cookie value directly to PHP's native unserialize() function with no input validation, type checking, or class whitelisting. An unauthenticated attacker supplies a crafted serialized PHP object in the cookie header; deserialization triggers magic methods (__wakeup, __destruct) on attacker-controlled class instances. Magento 2's dependency tree contains exploitable gadget chains, enabling arbitrary code execution under the web process account. MITRE maps this to T1190 (Exploit Public-Facing Application) and T1059.004 (Unix Shell). No

authentication or user interaction is required. EPSS score is 0.00104 (low statistical probability), but CISA's KEV listing confirms active real-world exploitation is already occurring - prioritize KEV status over statistical scores. The fix is a version upgrade to 1.11.12 or later.

Action Checklist

1. Step 1, Containment: Immediately identify all Magento 2 instances running Mirasvit Full Page Cache Warmer. If upgrading within 24 hours is not possible, configure your WAF or reverse proxy to inspect and block requests containing serialized PHP object patterns (O: followed by digit sequences) in the CacheWarmer cookie. Temporarily disabling the extension via Magento admin (Stores > Configuration > Mirasvit > Cache Warmer) removes the attack surface without taking the storefront offline. NIST AC-4 (Information Flow Enforcement) and CIS 4.4 (Implement and Manage a Firewall on Servers) directly support this step.
2. Step 2, Detection: Query web server access logs for requests carrying a CacheWarmer cookie value beginning with the PHP serialization prefix 'O:' or 'a:'. In your SIEM, search for HTTP requests with Set-Cookie or Cookie headers matching the pattern CacheWarmer=O%3A or CacheWarmer=a%3A. Review application logs under var/log/ for unexpected PHP fatal errors, unserialize() warnings, or new files written to pub/media or pub/static by the web process. Check for anomalous outbound connections from the web server process (e.g., wget, curl, bash spawned by php-fpm). NIST AU-6 (Audit Record Review, Analysis, and Reporting) and CIS 8.2 (Collect Audit Logs) are the governing controls. File integrity monitoring and local process monitoring capabilities apply directly.
3. Step 3, Eradication: Upgrade Mirasvit Full Page Cache Warmer to version 1.11.12 or later via Composer: run 'composer require mirasvit/module-cache-warmer:^1.11.12 --update-with-dependencies' in the Magento 2 root, then 'php bin/magento setup:upgrade && php bin/magento cache:flush'. Verify the installed version with 'composer show mirasvit/module-cache-warmer'. If compromise is suspected, do not patch in place; restore from a known-good backup taken before the earliest possible exploitation window and apply the patch to the restored environment. CIS 7.3 and 7.4 (Automated OS and Application Patch Management) govern this step.
4. Step 4, Recovery: After patching, revalidate the extension version, re-run a Magento integrity check ('php bin/magento module:status'), and scan pub/media, pub/static, and vendor directories for unexpected PHP files using a file integrity tool. Monitor web process outbound network activity and authentication logs for 72 hours post-remediation. Re-enable any WAF rules tuned for this CVE and confirm they remain active. NIST SI-4 (System Monitoring) and AU-12 (Audit Record Generation) govern post-recovery monitoring. File integrity monitoring remains applicable.
5. Step 5, Post-Incident: Conduct a gap review against NIST AC-6 (Least Privilege), confirm the web process account has no write access outside of necessary directories. Evaluate whether all third-party Magento extensions undergo security review before installation (CIS 2.2: Ensure Authorized Software is Currently Supported). Establish a process to monitor Mirasvit and other extension vendors for security advisories. Review WAF rulesets to ensure PHP deserialization attack patterns are covered at baseline. Document findings under your incident response playbook in accordance with NIST IR controls.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal counsel, and payment brand incident response (Visa/Mastercard) immediately if forensic analysis confirms any request with a successful HTTP 200 response carrying a CacheWarmer=O%3A cookie payload, any PHP file written to pub/media or pub/static by the web process, any rogue admin_user account created during the exposure window, or any evidence of outbound connections from php-fpm — as these indicators collectively suggest active exploitation of an internet-facing Magento 2 e-commerce environment, triggering PCI DSS Requirement 12.10.5 incident notification obligations and potentially state breach notification laws if cardholder or PII data was accessible.
Recovery Notes	After confirming Mirasvit Full Page Cache Warmer 1.11.12 is installed and verified via 'composer show mirasvit/module-cache-warmer', perform a full Magento cache flush and reindex to eliminate any attacker-controlled cached objects that may have been serialized into the cache store during the exploitation window — run 'php bin/magento cache:flush && php bin/magento indexer:reindex'. Monitor Nginx access logs and Magento var/log/exception.log continuously for 72 hours for recurring CacheWarmer deserialization patterns or new PHP fatal errors from the Mirasvit namespace, which would indicate a reinfection or missed webshell. If the storefront processes payment card data, engage your PCI QSA to assess whether a forensic investigation is required before returning to full production status under PCI DSS Requirement 12.10.
Forensic Artifacts	Nginx or Apache access logs filtered for HTTP requests with Cookie headers matching 'CacheWarmer=O%3A' or 'CacheWarmer=a%3A' — the URL-encoded PHP serialization prefixes that uniquely identify exploitation attempts against the Mirasvit Full Page Cache Warmer unserialize() endpoint; preserve raw log files with SHA-256 hashes immediately upon detection. | Magento application exception log at var/log/exception.log searched for 'unserialize()', class instantiation errors in the Mirasvit\CacheWarmer namespace, or PHP fatal errors coinciding with suspicious cookie activity — these entries record the server-side execution of the deserialization payload. | Filesystem listing of all .php and .phtml files in pub/media and pub/static with creation timestamps newer than the last known-good deployment — generated with 'find /var/www/html/pub/media /var/www/html/pub/static -type f \(-name "*.php" -o -name "*.phtml" \) -ls' — these directories are writable by the web process and are the primary webshell drop locations following successful RCE via this vulnerability. | Magento database admin_user table snapshot — 'SELECT user_id, username, email, created_in, created_at FROM admin_user ORDER BY created_at DESC;' — to identify backdoor administrator accounts created by an attacker who leveraged RCE to escalate into the Magento admin panel after initial exploitation. | Auditd or syslog records of child processes spawned by the php-fpm process (UID www-data) during the exploitation window — specifically bash, sh, wget, curl, python, or nc — which represent the post-exploitation command execution phase following successful PHP object injection deserialization via the CacheWarmer cookie.

Per-Action IR Details

Step 1 — Containment: Immediately identify all Magento 2 instances running Mirasvit Full Page Cache Warmer. If upgrading within 24 hours is not possible, configure your WAF or reverse proxy to inspect and block requests containing serialized PHP object patterns (O: followed by digit sequences) in the CacheWarmer cookie. Temporarily disabling the extension via Magento admin (Stores > Configuration > Mirasvit > Cache Warmer) removes the attack surface without taking the storefront offline. NIST AC-4 (Information Flow Enforcement) and CIS 4.4 (Implement and Manage a Firewall on Servers) directly support this step.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST IR-4 (Incident Handling), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without an enterprise WAF, deploy ModSecurity (free, open-source) on your Apache or Nginx reverse proxy with the OWASP Core Rule Set (CRS) rule 933160, which blocks PHP object injection patterns including 'O:' serialized payloads in cookies. Alternatively, use this nginx snippet in your server block: 'if ($http_cookie ~* "CacheWarmer=O%3A|CacheWarmer=a%3A") { return 403; }'. Confirm blocking is active by sending a test curl request: 'curl -H "Cookie: CacheWarmer=O%3A8%3A..." https://yourstore.com' and verifying a 403 response. Disable the module via CLI if admin access is unavailable: 'php bin/magento module:disable Mirasvit_CacheWarmer && php bin/magento cache:flush'.

Evidence: Before disabling the extension or modifying WAF rules, capture the following: (1) Full Nginx or Apache access logs from the last 30 days — 'cp /var/log/nginx/access.log /evidence/nginx_access_pre_containment.log' — preserving timestamps for requests with CacheWarmer cookie values; (2) A snapshot of the currently installed module version — 'composer show mirasvit/module-cache-warmer > /evidence/composer_version_pre_patch.txt'; (3) A directory listing with timestamps of pub/media, pub/static, and var/tmp — 'find /var/www/html/pub /var/www/html/var/tmp -type f -newer /var/www/html/composer.lock -ls > /evidence/new_files_pre_containment.txt' — to establish a baseline of files created after the last known-good deployment; (4) Current php-fpm or web process network connections — 'ss -tulnp | grep php-fpm > /evidence/network_state_pre_containment.txt'.

Step 2 — Detection: Query web server access logs for requests carrying a CacheWarmer cookie value beginning with the PHP serialization prefix 'O:' or 'a:'. In your SIEM, search for HTTP requests with Set-Cookie or Cookie headers matching regex pattern CacheWarmer=O%3A or CacheWarmer=a%3A. Review application logs under var/log/ for unexpected PHP fatal errors, unserialize() warnings, or new files written to pub/media or pub/static by the web process. Check for anomalous outbound connections from the web server process (e.g., wget, curl, bash spawned by php-fpm). NIST AU-6 (Audit Record Review, Analysis, and Reporting) and CIS 8.2 (Collect Audit Logs) are the governing controls. D3-SFA (System File Analysis) and D3-LAM (Local Account Monitoring) apply directly.

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-2 (Event Logging), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, run this grep command directly against Nginx or Apache access logs to surface all exploitation attempts against the CacheWarmer deserialization endpoint: 'grep -E "CacheWarmer=O%3A|CacheWarmer=a%3A|CacheWarmer=O:|CacheWarmer=a:" /var/log/nginx/access.log | awk '{print $1, $4, $7, $9}' > /evidence/cachewarmer_hits.txt'. For process-level detection of post-exploitation activity (e.g., php-fpm spawning bash or wget), install auditd and add this rule: 'auditctl -a always,exit -F arch=b64 -S execve -F ppid=$(pgrep php-fpm | head -1) -k php_child_exec'. Review output with 'ausearch -k php_child_exec'. Use 'find /var/www/html/pub/media /var/www/html/pub/static -name "*.php" -newer /var/www/html/composer.lock' to detect webshells written by the exploited web process.

Evidence: Preserve before analysis: (1) Unmodified raw access logs — 'cp /var/log/nginx/access.log /evidence/' and hash them — 'sha256sum /var/log/nginx/access.log > /evidence/access_log.sha256'; (2) Magento application exception and system logs — 'cp /var/www/html/var/log/exception.log /var/www/html/var/log/system.log /evidence/' — search specifically for 'unserialize()' warnings, 'PHP Fatal error', or class instantiation errors tied to the CacheWarmer module namespace (Mirasvit\CacheWarmer); (3) PHP-FPM slow log and error log — 'cp /var/log/php-fpm/error.log /evidence/php_fpm_error_pre.log' — entries showing abnormal execution time spikes during cookie processing indicate active exploitation; (4) Web server process open file descriptors at time of analysis — 'lsof -p $(pgrep php-fpm) > /evidence/php_fpm_open_files.txt' — to identify any webshells currently open by the process; (5) Outbound DNS and connection log — 'grep php-fpm /var/log/syslog > /evidence/syslog_php_fpm.txt' — to catch C2 beacon or payload retrieval activity post-exploitation.

Step 3 — Eradication: Upgrade Mirasvit Full Page Cache Warmer to version 1.11.12 or later via Composer: run 'composer update mirasvit/module-cache-warmer' in the Magento 2 root, then 'php bin/magento setup:upgrade && php bin/magento cache:flush'. Verify the installed version with 'composer show mirasvit/module-cache-warmer'. If compromise is suspected, do not patch in place; restore from a known-good backup taken before the earliest possible exploitation window and apply the patch to the restored environment. CIS 7.3 and 7.4 (Automated OS and Application Patch Management) govern this step.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: If Composer cannot be run in a locked-down environment, download the 1.11.12 release archive directly from the Mirasvit vendor portal or Magento Marketplace, extract it over the existing 'vendor/mirasvit/module-cache-warmer' directory, and run 'php bin/magento setup:upgrade && php bin/magento cache:flush' manually. Before patching, generate a file integrity baseline of the Magento installation with: 'find /var/www/html -type f -name "*.php" | xargs md5sum > /evidence/php_file_hashes_pre_patch.txt' — compare post-patch with 'diff /evidence/php_file_hashes_pre_patch.txt /evidence/php_file_hashes_post_patch.txt' to identify any attacker-planted files that survive the patch. If restoring from backup, verify backup integrity before restoration: 'sha256sum backup.tar.gz' against a stored reference hash.

Evidence: Before executing eradication, preserve a full forensic image of the compromised state: (1) Capture all PHP files in pub/media and pub/static that should never contain PHP — 'find /var/www/html/pub/media /var/www/html/pub/static -name "*.php" -o -name "*.phtml" > /evidence/suspicious_php_in_pub.txt' — these are likely webshells persisted by the attacker after exploiting the unserialize() vulnerability; (2) Dump all currently scheduled Magento cron jobs and system crontabs for the web user — 'crontab -u www-data -l > /evidence/crontab_www_data.txt' — attackers frequently add cron-based persistence after achieving RCE; (3) Capture the composer.lock and composer.json files before and after patching to document the exact version delta; (4) Collect a memory snapshot of php-fpm if active exploitation is suspected — 'gcore $(pgrep php-fpm | head -1)' — for post-incident analysis of in-memory payloads from the deserialization chain.

Step 4 — Recovery: After patching, revalidate the extension version, re-run a Magento integrity check ('php bin/magento module:status'), and scan pub/media, pub/static, and vendor directories for unexpected PHP files using a file integrity tool. Monitor web process outbound network activity and authentication logs for 72 hours post-remediation. Re-enable any WAF rules tuned for this CVE and confirm they remain active. NIST SI-4 (System Monitoring) and AU-12 (Audit Record Generation) govern post-recovery monitoring. D3-SFA (System File Analysis) remains applicable.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-4 (System Monitoring), NIST AU-12 (Audit Record Generation), NIST CP-10 (System Recovery and Reconstitution), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 8.2 (Collect Audit Logs)

Compensating: Without EDR for outbound monitoring, use auditd with the following rule to alert on any network connection initiated by the web process post-recovery: 'auditctl -a always,exit -F arch=b64 -S connect -F uid=$(id -u www-data) -k web_outbound'. Monitor output daily with 'ausearch -k web_outbound -ts today'. For file integrity monitoring of pub/media, pub/static, and vendor/mirasvit without a commercial FIM tool, schedule a cron job every 6 hours: 'find /var/www/html/pub /var/www/html/vendor/mirasvit -type f -name "*.php" | xargs sha256sum > /tmp/fim_check_$(date +%s).txt && diff /evidence/php_file_hashes_post_patch.txt /tmp/fim_check_$(date +%s).txt | mail -s "FIM Alert" security@yourorg.com'. Also run Magento's native integrity check: 'php bin/magento module:status | grep -i mirasvit' to confirm only version 1.11.12 is active.

Evidence: During the 72-hour monitoring window, collect and preserve: (1) Web server access logs in 24-hour segments — attacker reentry attempts will reuse CacheWarmer=O%3A cookie patterns; (2) Magento admin panel login logs — 'var/log/system.log' filtered for 'adminhtml' authentication events — attackers who achieved RCE may

have created rogue admin accounts; (3) Database audit of the admin_user table — 'SELECT user_id, username, email, created_at FROM admin_user ORDER BY created_at DESC LIMIT 20;' — to detect backdoor admin accounts created during the exploitation window; (4) Output of 'php bin/magento module:status' post-patch captured to file for verification record; (5) Network flow logs or iptables counters showing outbound connections from the web server IP during the recovery window.

Step 5 — Post-Incident: Conduct a gap review against NIST AC-6 (Least Privilege) — confirm the web process account has no write access outside of necessary directories. Evaluate whether all third-party Magento extensions undergo security review before installation (CIS 2.2: Ensure Authorized Software is Currently Supported). Establish a process to monitor Mirasvit and other extension vendors for security advisories. Review WAF rulesets to ensure PHP deserialization attack patterns are covered baseline. Document findings under your incident response playbook in accordance with NIST IR controls.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST RA-3 (Risk Assessment), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: To audit web process filesystem permissions without enterprise tooling, run: 'find /var/www/html -not -path "*/pub/media/*" -not -path "*/var/*" -not -path "*/generated/*" -writable -user www-data -type f > /evidence/excessive_write_permissions.txt' — any result outside of pub/media, var/, and generated/ represents a least-privilege violation that this CVE class could exploit for persistence. For vendor advisory monitoring without a commercial feed, create a free GitHub account and watch the Mirasvit GitHub repository (https://github.com/mirasvit) for release and security advisory notifications. Subscribe to the Magento Security Alerts mailing list at https://magento.com/security and configure a free RSS alert via Feedly for 'Mirasvit CVE' to catch future disclosures. Document the PHP deserialization attack pattern (O: cookie prefix) as a permanent WAF baseline rule, not a temporary CVE-specific block.

Evidence: For the post-incident lessons-learned record, compile: (1) The complete timeline reconstructed from Nginx access logs showing the first observed CacheWarmer=O%3A request versus the patch application timestamp — this establishes the exploitation window for breach notification purposes; (2) Output of the admin_user database query from Step 4 recovery, confirming whether any rogue accounts were created; (3) A list of all PHP files found in pub/media and pub/static during forensic analysis, with creation timestamps and SHA-256 hashes, to document the scope of any webshell deployment; (4) The composer.lock diff from pre- and post-patch eradication, documenting the exact Mirasvit module version change as evidence of remediation; (5) Documented WAF rule IDs added or modified specifically for CVE-2026-45247, to serve as a permanent configuration change record under NIST CM-3 (Configuration Change Control).

Detection Guidance

Primary detection signal: HTTP requests or server-side logs showing a CacheWarmer cookie value containing URL-encoded PHP serialization markers (O%3A, a%3A, O:, a:). In Apache or Nginx access logs, grep for 'CacheWarmer=O' or 'CacheWarmer=a'. In Splunk or similar SIEM, write a rule: index=web* (CacheWarmer=O%3A OR CacheWarmer=a%3A), adapt syntax to your SIEM platform. Secondary signals: new .php files created in pub/media, pub/static, or under vendor/ by the web server process (php-fpm, www-data, apache); unexpected child processes spawned by php-fpm (bash, sh, curl, wget, python); outbound connections from the web host to non-standard destinations on ports 4444, 1337, or ephemeral ranges. File integrity monitoring covers file-level indicators. Local process monitoring covers process and account anomalies. NIST AU-6 and AU-12 govern the log sources required to surface these signals. Note: EPSS score of 0.00104 reflects pre-KEV statistical probability; CISA KEV confirmation means active exploitation is occurring - weight detection findings accordingly.

Framework Mappings

MITRE-ATTACK

- **T1059.004** — Unix Shell
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **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

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

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
T1059.004	Unix Shell	Execution
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-45247	T1
CVE-2026-45247 | Mondoo Vulnerability Intelligence	https://mondoo.com/vulnerability-intelligence/vulnerability/CVE-202...	T3

Source	URL	Tier
CVE-2026-48247 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-48247	T1
CVE-2026-45247: Mirasvit Full Page Cache Remote codExecution	https://www.sherlockforensics.com/blog/2026-05-26-cve-2026-45247.html	T3
CVE-2026-45247 - Exploits & Severity - Feedly	https://feedly.com/cve/CVE-2026-45247	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-05-29 19:06 UTC by TJS Security Command Center