INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:41 UTC

PolyShell Mass Exploitation Targeting Magento 2 / Adobe Commerce, WebRTC Skimmer Deployed at Scale

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0106
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Magento Open Source 2 (all versions); Adobe Commerce (all versions without 2.4.9-beta1 patch, no stable-branch fix available as of disclosure)
Published	2026-03-25
Discovery Source	Rss

Executive Summary

An actively exploited critical vulnerability in Magento 2 and Adobe Commerce, tracked under Adobe bulletin APSB26-05 and dubbed 'PolyShell', has compromised approximately 56.7% of vulnerable storefronts since mass exploitation began March 19, 2026. Attackers are injecting a WebRTC-based payment card skimmer that is not restricted by standard Content Security Policy controls, resulting in real-time exfiltration of customer payment data from affected checkout flows. As of March 2026, the only available patch is a beta release (v2.4.9-beta1), raising the business risk for any organization running Magento 2 or Adobe Commerce in production.

Technical Analysis

PolyShell (APSB26-05) affects all versions of Magento Open Source 2 and Adobe Commerce without the v2.4.9-beta1 patch applied. As of March 2026 disclosure, no stable-branch remediation is available. The vulnerability cluster spans four CWEs: unrestricted file upload (CWE-434), code injection (CWE-94), cross-site scripting (CWE-79), and protection mechanism failure (CWE-693). Exploitation chain maps to MITRE ATT&CK T1190 (Exploit Public-Facing Application), T1505.003 (Server Software Component: Web Shell), T1059/T1059.007 (Command and Script Interpreter: JavaScript), T1056.003 (Input Capture: Web Portal Capture), T1071.001 (Application Layer Protocol: Web), T1027 (Obfuscated Files or Information), and T1036 (Masquerading). The novel payload uses WebRTC data channels as an exfiltration vector; WebRTC connections are not restricted by standard CSP unless explicitly configured. CVE identifier not confirmed in NVD

or CISA KEV as of item disclosure. Severity rating reflects the operational impact and active exploitation scope, not attribution confidence. Attribution details should be treated as medium confidence pending official registry confirmation. Source quality score: 0.56 (T3 sources only).

Action Checklist

1. Step 1, Patch decision (immediate): Evaluate deploying Magento Open Source 2.4.9-beta1 against the risk of running beta code in production. Given the active exploitation rate (approximately 56.7% of vulnerable stores), deploying the beta patch may be the lower-risk option compared to remaining unpatched. Confirm with your change management authority before deploying to production.
2. Step 2, Compensating controls (immediate, if patch is delayed): Block unauthenticated file upload endpoints at the WAF layer. Restrict admin panel access to known IP ranges. Enforce strict Content-Security-Policy headers and audit current CSP rules, noting that WebRTC connections ('connect-src' and 'default-src' directives) require explicit policy coverage to restrict.
3. Step 3, Inventory and exposure assessment (within 24 hours): Identify all Magento 2 and Adobe Commerce instances across production, staging, and dev environments. Confirm version numbers and patch status. Flag any internet-facing checkout flows for priority review.
4. Step 4, Skimmer detection sweep (within 24 hours): Review all JavaScript loaded on checkout and payment pages, compare against known-good baselines. Audit server-side files for unauthorized additions, particularly in media/upload directories and pub/static paths. Check for anomalous WebRTC connection attempts in network logs.
5. Step 5, Stakeholder notification and escalation: If any evidence of compromise is found, escalate to incident response immediately. Notify payment card brands and acquiring bank per PCI DSS breach notification obligations. Brief executive leadership given confirmed enterprise-level targeting. Update threat intelligence feeds with any confirmed IOCs.
6. Step 6, Long-term control review: Conduct a post-incident or post-patch review of file upload controls, CSP policy coverage (including WebRTC), and third-party JavaScript inventory. Add WebRTC exfiltration to detection rule sets. Monitor CISA KEV and NVD for official CVE assignment and update remediation records accordingly.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive leadership and IR team if any of the following are confirmed: (1) successful WebRTC skimmer deployment detected on production checkout; (2) evidence of exfiltrated payment data (network logs showing STUN/TURN outbound connections during card entry, or unauthorized data egress); (3) inability to patch within 48 hours due to change control blockers; (4) multiple Magento instances compromised across different business units or geographies, indicating coordinated campaign impact.

Recovery Notes	Post-containment: (1) rotate all admin credentials and API tokens, force re-authentication for active sessions, and review admin access logs from March 19, 2026 onward for unauthorized activity; (2) engage forensic analysis to determine exact compromise timeline and whether customer data was exfiltrated — preserve network captures and server logs for 90 days minimum per PCI-DSS requirement; (3) issue customer notification if payment data compromise is confirmed, coordinate with acquiring bank and card brands on breach notification process, and monitor fraud/chargeback patterns for 180 days post-incident; (4) update incident response runbook with PolyShell-specific detection procedures and WebRTC monitoring, and conduct post-incident review with all involved teams within 5 business days.
Forensic Artifacts	Magento application version log (bin/magento --version output, var/log/system.log entries with version strings) | Web server access logs (Apache access.log, Nginx access.log) filtered for file upload endpoints (media/upload/, pub/static/) from March 19, 2026 onward | PHP error logs (var/log/exception.log, var/log/system.log) showing file write operations, permission changes, or security-related errors | JavaScript source files on checkout page (pub/static/js/, media/upload/) with modification timestamps and content hashes (md5sum, sha256sum) compared to baseline | Network packet capture (tcpdump, Wireshark) of checkout flow traffic showing DNS queries, outbound connections to non-approved domains, and WebRTC signaling (STUN/TURN, wss://) from March 19, 2026 onward | Web application firewall or reverse proxy logs showing blocked/allowed requests to upload endpoints, admin panel, and checkout paths | Database transaction logs (if available) showing unauthorized schema modifications, user privilege changes, or backdoor account creation | Composer lock file (composer.lock) documenting third-party library versions and sources, compared to known-safe baseline | CSP violation reports (if CSP-Report-Only header is in place) showing attempted inline script execution or cross-origin resource loads

Per-Action IR Details

Step 1 — Patch decision (immediate): Evaluate deploying Magento Open Source 2.4.9-beta1 against the risk of running beta code in production. For most organizations, the active exploitation rate (~56.7% of vulnerable stores) makes applying the beta patch the lower-risk option. Confirm with your change management authority before deploying to production.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (preparation phase: vulnerability management and patch planning)

Controls: NIST 800-53 SI-2 (Flaw Remediation), CIS Controls 3.3 (Address Unauthorized Software)

Compensating: If beta deployment is blocked by change control: (1) snapshot production database and file system before any patching attempt; (2) document current version with `php -v` and `grep -r 'Magento\\Framework\\AppVersion' magento/` to establish baseline; (3) prepare rollback procedure: export database backup (`mysqldump -u root -p magento_db > backup_$(date +%s).sql`), tar current codebase (`tar -czf magento_backup_$(date +%s).tar.gz ./magento`), and test restoration in staging before production patch window.

Evidence: Before patching: capture current application version string (`bin/magento --version`), installed modules list (`bin/magento module:status > modules_baseline.txt`), file modification timestamps in `pub/static/` and `media/upload/` directories (`find pub/ media/ -type f -newermt '2026-03-19' -ls > recent_files.txt`), and database schema checksums (`mysqldump magento_db --no-data | md5sum > schema_baseline.txt`).

Step 2 — Compensating controls (immediate, if patch is delayed): Block unauthenticated file upload endpoints at the WAF layer. Restrict admin panel access to known IP ranges. Enforce strict Content-Security-Policy headers and audit current CSP rules, noting that WebRTC ('connect-src' and 'default-src' directives covering WebRTC/TURN) requires explicit policy coverage.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.2 (containment strategies: segmentation and access control)

Controls: NIST 800-53 AC-2 (Account Management), AC-3 (Access Enforcement), NIST 800-53 SI-4 (Information System Monitoring), CIS Controls 5.3 (Configure Data Access Control), 7.1 (Establish and Maintain a Data Security and Handling Policy)

Compensating: No SIEM/WAF environment: (1) at reverse proxy or web server layer (Apache/Nginx), block all POST requests to `media/upload/`, `pub/static/`, and admin URIs except from whitelisted IPs using ModRewrite or Nginx location blocks (`nginx: location /media/upload { deny all; } location /admin { allow 203.0.113.0/24; deny all; }`); (2) enforce CSP headers in `.htaccess` or Nginx config: `add_header Content-Security-Policy "default-src 'self'; connect-src 'self'; script-src 'self' 'nonce-RANDOM'; object-src 'none'; frame-ancestors 'none';"` (remove WebRTC defaults, explicitly exclude `wss://` and TURN server URIs); (3) log all blocked requests to dedicated file: `apache: CustomLog logs/blocked_uploads.log combined` for manual triage.

Evidence: Pre-implementation baseline: export current Apache/Nginx config (`cp /etc/apache2/sites-enabled/*.conf config_baseline_$(date +%s).tar.gz`), capture current CSP headers (`curl -i https://checkout.example.com | grep -i 'content-security-policy' > csp_baseline.txt`), log current file upload activity (`find media/upload/ -type f -mtime -1 -ls > upload_activity_baseline.txt`), and record admin access patterns (`grep -r 'admin' access.log | tail -1000 > admin_access_baseline.txt`).

Step 3 — Inventory and exposure assessment (within 24 hours): Identify all Magento 2 and Adobe Commerce instances across production, staging, and dev environments. Confirm version numbers and patch status. Flag any internet-facing checkout flows for priority review.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.1 (detection and analysis: scope determination and asset identification)

Controls: NIST 800-53 CM-2 (Baseline Configuration), CM-8 (Information System Component Inventory), CIS Controls 1.1 (Establish and Maintain Detailed Asset Inventory), 2.1 (Establish and Maintain Software Inventory)

Compensating: Without asset management tools: (1) query DNS and network device logs for known Magento hostnames (`dig +short *.magento.example.com`, `nslookup -type=any example.com | grep -i magento`); (2) scan internal network for common Magento paths (`curl -s http://internal-host:80/magento/bin/magento --version 2>/dev/null || echo 'not found'` across subnet); (3) manually audit application server configs: `ps aux | grep -i php`, `grep -r 'MAGENTO' /etc/ ~/.bashrc`, `ls -la /var/www/*/` for install directories; (4) create inventory spreadsheet with hostname, IP, version (from `bin/magento --version`), patch date, checkout domain, and internet accessibility.

Evidence: Capture for each instance: version output (`bin/magento --version > version_$(hostname).txt`), installed security patches (`grep -i 'security\|patch' var/log/system.log > patches_$(hostname).txt`), composer.lock dependencies (`cat composer.lock | jq '.packages[] | select(.name | contains("magento")) | {name, version}' > dependencies_$(hostname).json`), checkout page headers to confirm Magento signature (`curl -I https://checkout.example.com | head -20 > checkout_headers_$(date +%s).txt`), and network routing to confirm internet exposure (`traceroute example.com, nmap -sV checkout.example.com`).

Step 4 — Skimmer detection sweep (within 24 hours): Review all JavaScript loaded on checkout and payment pages — compare against known-good baselines. Audit server-side files for unauthorized additions, particularly in media/upload directories and pub/static paths. Check for anomalous WebRTC connection attempts in network logs.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.1 (detection and analysis: evidence acquisition and log analysis)

Controls: NIST 800-53 SI-4 (Information System Monitoring), AU-2 (Audit Events), SC-7 (Boundary Protection), CIS Controls 6.2 (Ensure Authorized Software is Currently Installed), 8.3 (Address Unauthorized Software)

Compensating: Without EDR/SIEM: (1) baseline checkout JavaScript: `curl -s https://checkout.example.com | grep -oP '(?]*src="')[^"]+' > checkout_scripts_baseline.txt`; compare against historical baseline or request-list to identify new includes; (2) audit file system for recent uploads: `find media/upload/ -type f -newermt '2026-03-19 00:00:00' -exec ls -lah {} \; > recent_media_files.txt`; check for `.js`, `.json`, `.wasm` anomalies; (3) search pub/static for modified files: `find pub/static/ -type f -newermt '2026-03-19' | xargs grep -l 'WebRTC\|RTCPeerConnection\|stun:\|turn:' > potential_skimmers.txt`; (4) manually inspect network logs (tcpdump or firewall logs) for outbound STUN/TURN traffic: `grep -i 'stun\|turn\|wss://\|rtc' firewall.log | grep -v internal > anomalous_connections.txt`.

Evidence: Preserve before analysis: full HTML source of checkout page (`curl -s https://checkout.example.com > checkout_page_$(date +%s).html`), all external script sources and their content (`curl -s [script-url] > script_$(basename $url)_$(date +%s).js` for each external script), JavaScript source maps if available (`curl -s checkout.js.map > checkout_map_$(date +%s).json`), file modification times and hashes for media/upload and pub/static (`find media/ pub/static/ -type f -exec md5sum {} \; > file_hashes_$(date +%s).txt`), and network packet capture during checkout flow (`tcpdump -i eth0 'host example.com' -w checkout_traffic_$(date +%s).pcap`).

Step 5 — Stakeholder notification and escalation: If any evidence of compromise is found, escalate to incident response immediately. Notify payment card brands and acquiring bank per PCI DSS breach notification obligations. Brief executive leadership given confirmed enterprise-level targeting. Update threat intelligence feeds with any confirmed IOCs.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.1 (detection and analysis: incident declaration and notification), NIST 800-61r3 §3.4 (post-incident activities: lessons learned)

Controls: NIST 800-53 IR-2 (Incident Response Training), IR-4 (Incident Handling), IR-6 (Incident Reporting), CIS Controls 19.1 (Establish and Maintain an Incident Response Process)

Compensating: For organizations without formal IR function: (1) document evidence immediately: create incident log with datetime, compromised systems, IOCs, and evidence file paths (`echo "Incident: PolyShell, Time: $(date -u +%Y-%m-%dT%H:%M:%SZ), Systems: [list], Evidence: [paths]" >> incident_log_$(date +%s).txt`); (2) notify payment processor via documented breach notification process (reference your PCI-DSS attestation document for processor contact and timeline — typically 30 days); (3) engage external IR firm if internal capability is insufficient — request forensic preservation protocol before they begin (`Chain-of-Custody form, evidence inventory, forensic image methodology`); (4) publish IOCs to CISA AIS portal or threat intelligence sharing platform (e.g., AlienVault OTX) with anonymization if needed.

Evidence: Document notification chain: (1) incident declaration form with date/time, affected systems, compromise indicators, and decision authority; (2) communication log with timestamps of all notifications (IR team, management, payment processor, legal); (3) evidence inventory spreadsheet listing all collected artifacts (file paths, hashes, capture times); (4) IOC export (IPs, domains, file hashes, file paths) formatted for threat intel platform submission (`json: {"type": "file_hash", "value": "abc123...", "context": "PolyShell skimmer", "date": "2026-03-04"}`).

Step 6 — Long-term control review: Conduct a post-incident or post-patch review of file upload controls, CSP policy coverage (including WebRTC), and third-party JavaScript inventory. Add WebRTC exfiltration to detection rule sets. Monitor CISA KEV and NVD for official CVE assignment and update remediation records accordingly.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §3.4 (post-incident activities: lessons learned and improved processes)

Controls: NIST 800-53 SI-2 (Flaw Remediation), IR-7 (System Monitoring), CA-7 (Continuous Monitoring), CIS Controls 3.3 (Address Unauthorized Software), 6.2 (Ensure Authorized Software), 19.1 (Establish and Maintain Incident Response Process)

Compensating: Without enterprise monitoring: (1) schedule quarterly file upload control audits: document all upload endpoints, authentication requirements, file type restrictions, and storage location (`audit_checklist=$(cat /etc/csp_policy.txt`), require change request and approval before modification, and test in staging with Content-Security-Policy-Report-Only header first; (3) inventory third-party JavaScript monthly: extract all external script sources (`curl -s https://checkout.example.com | grep -oP '(?]*src=")[^"]+' > js_inventory_$(date +%Y-%m).txt`), version-lock libraries, and audit for known vulnerabilities (`npm audit`, `composer audit` for PHP deps); (4) create detection rule for WebRTC exfiltration: log and alert on outbound `wss://` connections, `RTCPeerConnection` constructor calls, and STUN/TURN server connections — implement via web application firewall or application logging (`grep -i 'RTCPeerConnection\|stun:\|turn:' app.log`).

Evidence: Archive post-incident: (1) lessons-learned report (what happened, why, how to prevent recurrence, action items with owners and deadlines); (2) updated control documentation: file upload policy (`latest_upload_policy_v2.txt` with approved endpoint list and validation rules), CSP policy file (`csp_policy_production_v2.txt` with WebRTC

restrictions), third-party JS inventory (`approved_js_libraries_2026-Q2.csv` with versions and audit dates); (3) detection rules in SIEM/WAF format (`.yml` for Sigma rules, ModSecurity rules file, or IDS rulesets); (4) remediation tracking spreadsheet with step number, action, responsible party, due date, completion date, and evidence link.

Detection Guidance

Primary indicators to investigate: (1) New or modified PHP files in Magento upload directories (pub/media, var/import, app/code), compare file hashes against deployment baseline or last known-good state. (2) Unexpected JavaScript inclusions on checkout pages, diff current page source against a stored baseline; flag any script tags referencing external domains or encoded payloads. (3) WebRTC connection attempts from checkout pages, review browser-side network telemetry or proxy logs for RTCPeerConnection establishment or STUN/TURN traffic originating from storefront pages; standard CSP does not block WebRTC by default. (4) Anomalous POST requests to admin or API endpoints suggesting file upload exploitation, look for multipart/form-data requests to non-standard paths in web server access logs. (5) Outbound HTTP/S connections from web server processes to unfamiliar external IPs, check for C2 beaconing patterns consistent with T1071.001. (6) Web shell artifacts, scan for small PHP files with eval(), base64_decode(), or system() calls in web-accessible directories. Behavioral hunting hypothesis: a checkout session that initiates a WebRTC peer connection to an external IP with no corresponding legitimate third-party service is a strong indicator of active skimmer operation. Note: no confirmed IOCs (hashes, domains, IPs) are available in the current source data, detection must rely on behavioral and integrity-based methods until IOCs are published by a primary authority such as CISA or a reputable threat intelligence provider.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`No confirmed IOCs available`	No specific IP addresses, domains, file hashes, or URLs have been confirmed by a primary or secondary authority source as of item disclosure. Do not use unverified IOCs from social media or unattributed reports. Monitor CISA, NVD, and reputable threat intelligence feeds for published indicators.	LOW

Framework Mappings

MITRE-ATTACK

- **T1059.007** — JavaScript
- **T1190** — Exploit Public-Facing Application
- **T1056.003** — Web Portal Capture
- **T1059** — Command and Scripting Interpreter
- **T1071.001** — Web Protocols
- **T1505.003** — Web Shell

- **T1027** — Obfuscated Files or Information
- **T1036** — Masquerading

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
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-2** — Baseline Configuration
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A03:2021** — Injection

CIS-V8

- **16.10**
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1059.007	JavaScript	Execution
T1190	Exploit Public-Facing Application	Initial-Access
T1056.003	Web Portal Capture	Collection
T1059	Command and Scripting Interpreter	Execution
T1071.001	Web Protocols	Command-And-Control
T1505.003	Web Shell	Persistence
T1027	Obfuscated Files or Information	Defense-Evasion

Technique ID	Technique Name	Tactic
T1036	Masquerading	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/polyshell-attacks-ta...	T3
Adobe Security Bulletin	https://helpx.adobe.com/security/products/magento/apsb26-05.html	T3
Adobe Releases Security Patch for Adobe Commerce and Magento	https://www.linkedin.com/posts/atwix_adobe-just-released-a-critical...	T3
Adobe Commerce and Magento Open Source (APSB26-05)	https://magecomp.com/blog/security-update-adobe-commerce-magento-op ...	T3
Magento Open Source 2.4.9-beta1 release notes | Adobe Commerce	https://experienceleague.adobe.com/en/docs/commerce-operations/rele...	T3

Generated 2026-03-29 18:41 UTC by TJS Security Command Center