INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-06 14:05 UTC

Polyfill.io CDN Reactivation Triggers Credential-Harvesting Login Prompts on Major Brand Sites

THREAT CAMPAIGN | **MEDIUM** | CVSS 5.0

SCC Item ID	SCC-CAM-2026-0420
Type	Threat Campaign
Severity	MEDIUM
CVSS Base Score	5.0
Affected Products	Polyfill.io CDN (reactivated ~May 2026); affected downstream sites include Toshiba, Muji, Samsung Smart TVs, Zojirushi, FiNC Technologies, Ishiyaku Publishers, Hobonichi
Published	2026-06-05T17:54:42
Discovery Source	Rss

Executive Summary

The polyfill[.]io CDN domain, compromised in early 2024 when acquired by Funnull CDN, a Chinese entity, reactivated in late May 2026 and began serving HTTP 401 responses that trigger browser-native credential prompts on any site still loading scripts from the domain. Confirmed affected sites include Toshiba, Muji, Samsung Smart TV portal, Zojirushi, and several Japanese consumer brands. The business risk is direct: users visiting these sites see authentic-looking browser login dialogs, creating a convincing credential-harvesting surface with no malicious code on the affected site itself, making detection and attribution difficult for both site owners and end users.

Technical Analysis

Attack vector: supply chain abuse via third-party CDN dependency (CWE-829: Inclusion of Functionality from Untrusted Control Sphere; CWE-1104: Use of Unmaintained Third-Party Components; CWE-494: Download of Code Without Integrity Check). The polyfill[.]io domain, post-acquisition by Funnull CDN, returned HTTP 401 Unauthorized responses upon reactivation in late May 2026. Any browser loading a script tag referencing polyfill[.]io received a 401, which triggers the browser's native WWW-Authenticate dialog, a credential prompt that appears to originate from the visited site, not an external domain. Relevant MITRE ATT&CK techniques: T1195.002 (Compromise Software Supply Chain), T1056.003 (Web Portal Capture), T1059.007 (JavaScript execution), T1189 (Drive-by Compromise), T1584.001 (Compromised Infrastructure), T1071.001 (Web Protocols for C2), T1556 (Modify Authentication Process). No CVE has been assigned. No confirmed credential

exfiltration reported as of reporting date. Affected sites do not contain malicious code directly; the attack surface exists entirely through the external script dependency. The 2024 original compromise affected an estimated 100,000+ sites (per Sonatype reporting); the 2026 reactivation affects the subset that never removed the dependency.

Action Checklist

1. Step 1: Containment. Audit all production web properties for script tags, dynamic imports, or iframe src attributes referencing polyfill[.]io. Block outbound requests to polyfill[.]io at the WAF, proxy, or DNS layer immediately. Do not wait for a vendor advisory, no patch exists; removal of the dependency is the fix. Confirm third-party script dependencies are tracked in your software inventory (CIS 2.1). Extend inventory scope to include front-end dependencies and CDN references. Verify subresource integrity (SRI) attributes are present and correctly configured on any remaining third-party script tags.
2. Step 2: Detection. Search web server access logs and CDN logs for outbound requests to polyfill[.]io from any production asset. Query SIEM for HTTP 401 responses originating from polyfill[.]io in browser telemetry or proxy logs. In browser-side monitoring (if deployed), look for WWW-Authenticate challenge events correlated to script load failures. IOC: domain polyfill[.]io; any script src or dynamic fetch targeting cdn.polyfill.io or polyfill.io/* should be treated as compromised. Reference NIST AU-2 (Event Logging), confirm logging captures outbound script load requests (NIST AU-6: Audit Record Review, Analysis, and Reporting).
3. Step 3: Eradication. Remove all references to polyfill[.]io from source code, build configurations, CMS templates, and third-party tag manager entries. Replace with either: (a) the Cloudflare-maintained fork at cdnjs.cloudflare.com/polyfill (verify current availability independently), (b) a self-hosted copy pinned to a verified commit, or (c) eliminate the dependency entirely, modern browsers cover most polyfill use cases natively. Enforce subresource integrity (SRI) hashes on all remaining third-party scripts. Reference CIS 2.2 (Ensure Authorized Software is Currently Supported) and CIS 2.3 (Address Unauthorized Software) applied to front-end dependency inventory.
4. Step 4: Recovery. After removing the polyfill[.]io reference, verify remediation by loading affected pages in an isolated browser and confirming no 401 dialogs appear. Run a fresh dependency scan against all production URLs. Monitor proxy and DNS logs for 48-72 hours post-remediation to confirm no residual calls to polyfill[.]io (e.g., from cached builds, CDN edge caches, or third-party tag managers not yet updated). Monitor for anomalous authentication events from users of affected pages during the exposure window (NIST AU-6: Audit Record Review).
5. Step 5: Post-Incident. Conduct a full third-party script inventory across all web properties; document every externally hosted script with source, purpose, and owner (CIS 1.1: Establish and Maintain Detailed Enterprise Asset Inventory, extend scope to include front-end dependencies). Implement a Content Security Policy (CSP) that restricts script-src to an allowlist of trusted origins, blocking unauthorized external script loads at the browser level. Establish a recurring review process for third-party CDN dependencies (CIS 7.1: Establish and Maintain a Vulnerability Management Process). This incident is a direct consequence of CWE-1104 (unmaintained third-party components), the 2024 compromise was widely publicized; lack of remediation over two years represents a governance failure addressable through NIST AC-20 (Use of External Systems) policy enforcement.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to executive leadership and legal/privacy counsel immediately if proxy or IdP authentication logs confirm any user credentials were submitted in response to the Funnull CDN HTTP 401 browser prompt on affected consumer-facing properties, as this constitutes a credential-harvesting event with potential PII exposure triggering breach notification obligations under GDPR, CCPA, or applicable Japanese privacy law (Act on the Protection of Personal Information) depending on affected user base.
Recovery Notes	After removing all polyfill[.]io references and purging CDN edge caches, monitor proxy and DNS resolver logs continuously for 72 hours to detect residual calls originating from third-party tag manager containers, A/B testing tools, or marketing pixels that may independently load polyfill[.]io without appearing in your primary source code. Verify that the replacement script source — whether self-hosted or Cloudflare cdnjs — is loading correctly across the browser matrix your user base requires, since the original polyfill dependency existed for a reason. If any authentication anomalies are identified during the exposure window review, treat affected user accounts as potentially compromised and initiate a forced credential reset workflow before declaring full recovery.
Forensic Artifacts	Browser HAR files captured from affected production pages (Toshiba, Muji, Samsung Smart TV portal, etc.) showing the polyfill[.]io script GET request, the HTTP 401 response from Funnull CDN infrastructure, and the WWW-Authenticate header that triggered the native browser credential dialog | Web server and CDN access logs (Apache/Nginx access.log, CloudFront/Cloudflare access logs) filtered for requests to or referencing polyfill[.]io from approximately May 20, 2026 onward, preserving client IP, timestamp, User-Agent, HTTP status code, and referrer fields | DNS resolver query logs showing polyfill[.]io and cdn.polyfill.io resolution requests from production server IPs, including the resolved IP addresses (documenting Funnull CDN infrastructure) and query timestamps to establish the exposure window | Identity provider (IdP) or web application authentication logs covering the exposure window, filtered for any login events, failed authentication attempts, or session creation events correlated to user activity on the affected brand sites, to assess whether credentials submitted to the spoofed 401 prompt resulted in downstream account access | Third-party tag manager container export files (Google Tag Manager JSON export, Adobe Launch rule export) from each affected property, preserved as-found before eradication, documenting the polyfill[.]io reference within tag manager rules that would not appear in static source code audits

Per-Action IR Details

Step 1: Containment — Audit all production web properties for script tags, dynamic imports, or iframe src attributes referencing polyfill[.]io. Block outbound requests to polyfill[.]io at the WAF, proxy, or DNS layer immediately. Do not wait for a vendor advisory — no patch exists; removal of the dependency is the fix. Verify subresource integrity (SRI) attributes are absent or misconfigured on any remaining third-party script tags (per CIS 2.1: Establish and Maintain a Software Inventory — extend to front-end dependencies).

NIST Phase: Containment

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

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: For teams without a WAF: add a DNS sinkhole entry for polyfill[.]io and cdn.polyfill.io in your internal resolver (bind/unbound config or Windows DNS RPZ) pointing to 0.0.0.0. Simultaneously, run a grep across all web root directories: `grep -rn 'polyfill\.io' /var/www/ --include='*.html' --include='*.js' --include='*.php' --include='*.twig'`. For tag manager entries (Google Tag Manager, Adobe Launch), log into each console and search custom HTML tags for

'polyfill.io' — these are frequently missed in source-code audits. For proxy-based blocking without enterprise tooling, add a squid ACL deny rule or iptables OUTPUT DROP rule targeting the resolved IP ranges of polyfill[.]io.

Evidence: Before blocking, capture a full HTTP archive (HAR file) from an affected production page using browser DevTools (Network tab → Export HAR) to document the polyfill[.]io script load request, the resulting HTTP 401 response, the WWW-Authenticate header value served by Funnull CDN, and any credential prompt metadata. Preserve this HAR as forensic evidence of the active credential-harvesting mechanism. Also snapshot current DNS resolution for polyfill[.]io and cdn.polyfill.io (`dig polyfill.io +short` / `nslookup cdn.polyfill.io`) to document Funnull CDN infrastructure at the moment of containment.

Step 2: Detection — Search web server access logs and CDN logs for outbound requests to polyfill[.]io from any production asset. Query SIEM for HTTP 401 responses originating from polyfill[.]io in browser telemetry or proxy logs. In browser-side monitoring (if deployed), look for WWW-Authenticate challenge events correlated to script load failures. IOC: domain polyfill[.]io; any script src or dynamic fetch targeting cdn.polyfill.io or polyfill.io/* should be treated as compromised. Reference NIST AU-2 (Event Logging) — confirm logging captures outbound script load requests (NIST AU-6: Audit Record Review, Analysis, and Reporting).

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 AU-3 (Content Of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Without SIEM: run the following against Apache/Nginx access logs to identify polyfill.io script load requests and 401 responses: `grep -E 'polyfill\.io' /var/log/nginx/access.log | awk '{print $1, $7, $9}' | grep ' 401'`. For proxy logs (Squid): `grep 'polyfill.io' /var/log/squid/access.log | grep 'TCP_MISS'`. For sites behind Cloudflare or AWS CloudFront, pull the last 30 days of CDN access logs from the respective console and filter on the polyfill.io referrer or script-src hostname. To detect whether users received the WWW-Authenticate prompt, query web server error logs for 401 responses during the reactivation window (~May 2026 onward): `awk '$9==401' /var/log/apache2/access.log | grep -v 'localhost'`. Use Wireshark or tcpdump on the web server's outbound interface to capture live script fetch traffic: `tcpdump -i eth0 -w polyfill_capture.pcap host polyfill.io`.

Evidence: Collect proxy or forward-proxy logs showing outbound GET requests from production server IPs to polyfill[.]io with HTTP 200 responses (script delivery) followed by browser-side HTTP 401 (credential challenge) — this two-step sequence is the specific fingerprint of the Funnull CDN credential-harvesting mechanism. Preserve web server access logs with full timestamp, client IP, referrer, and user-agent fields intact (per NIST AU-3) covering the period from ~May 20, 2026 to present. If RUM (Real User Monitoring) or CSP reporting endpoints are deployed, extract any CSP violation reports citing polyfill[.]io as a blocked or flagged origin, which would document the exposure window and affected user sessions.

Step 3: Eradication — Remove all references to polyfill[.]io from source code, build configurations, CMS templates, and third-party tag manager entries. Replace with either: (a) the Cloudflare-maintained fork at cdnjs.cloudflare.com/polyfill (verify current availability independently), (b) a self-hosted copy pinned to a verified commit, or (c) eliminate the dependency entirely — modern browsers cover most polyfill use cases natively. Enforce subresource integrity (SRI) hashes on all remaining third-party scripts. Reference CIS 2.2 (Ensure Authorized Software is Currently Supported) and CIS 2.3 (Address Unauthorized Software) applied to front-end dependency inventory.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 2.1 (Establish and Maintain a Software Inventory), NIST AC-20 (Use Of External Systems)

Compensating: For teams without a dedicated dependency scanning tool: use `grep -rn 'polyfill\.io' ./ --include='*.json' --include='*.html' --include='*.js' --include='*.ts' --include='*.vue' --include='*.hbs'` across all repository roots to enumerate every reference. For CMS environments (WordPress, Drupal): also search the database for polyfill.io

references in widget/option tables — `SELECT option_name, option_value FROM wp_options WHERE option_value LIKE '%polyfill.io%'`. Validate SRI enforcement using the free srihash.org tool (verify current availability independently) to generate correct integrity= attributes for any replacement CDN-hosted scripts. For the self-hosted replacement path, pin to a specific Git commit hash from the polyfill.io GitHub archive and document the commit SHA in your software inventory per CIS 2.1.

Evidence: Before committing eradication changes, capture a diff of every modified file (git diff HEAD > polyfill_eradication_changes.patch) and preserve it as an eradication record. Document all tag manager container versions that contained the polyfill.io reference, including the GTM/Adobe Launch container ID, version number, and publish timestamp, to establish the full exposure timeline. If a self-hosted replacement is deployed, record the source commit SHA, download timestamp, and SHA-256 hash of the polyfill bundle file as chain-of-custody evidence that the replacement was not itself tampered with.

Step 4: Recovery — After removing the polyfill[.]io reference, verify remediation by loading affected pages in an isolated browser and confirming no 401 dialogs appear. Run a fresh dependency scan against all production URLs. Monitor proxy and DNS logs for 48–72 hours post-remediation to confirm no residual calls to polyfill[.]io (e.g., from cached builds, CDN edge caches, or third-party tag managers not yet updated). Reference NIST AU-6 (Audit Record Review) and NIST SI-4 (no mapped control — SI-4 not included in the provided knowledge base; do not cite) — monitor for anomalous authentication events from users of affected pages during the exposure window.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Verification without enterprise tooling: load each previously affected production URL in a clean browser profile (no cached resources) using Firefox with network logging enabled, or use `curl -v --max-time 10 https://[affected-site] 2>&1 | grep -i 'polyfill'` to confirm the domain no longer appears in page source or response headers. For CDN edge cache purging: issue a cache purge via the provider's CLI (e.g., `aws cloudfront create-invalidation --distribution-id XXXXX --paths '/*'` for CloudFront) and document the invalidation ID. Monitor DNS resolver logs for 48–72 hours using `journalctl -u named | grep polyfill.io` or equivalent to catch any residual resolution attempts from edge nodes or third-party tag managers not yet republished.

Evidence: During the 48–72 hour monitoring window, collect and preserve proxy or DNS logs showing zero resolution attempts to polyfill[.]io as positive evidence of successful containment — these serve as the remediation verification record. Separately, review authentication logs on any identity provider (IdP) or login system integrated with affected sites for anomalous credential submission events or unusual login source IPs during the exposure window (~May 20, 2026 to remediation date), as the HTTP 401 browser-native prompt may have caused users to submit credentials that were intercepted by Funnull CDN infrastructure.

Step 5: Post-Incident — Conduct a full third-party script inventory across all web properties; document every externally hosted script with source, purpose, and owner (CIS 1.1: Establish and Maintain Detailed Enterprise Asset Inventory — extend scope to include front-end dependencies). Implement a Content Security Policy (CSP) that restricts script-src to an allowlist of trusted origins, blocking unauthorized external script loads at the browser level. Establish a recurring review process for third-party CDN dependencies (CIS 7.1: Establish and Maintain a Vulnerability Management Process). This incident is a direct consequence of CWE-1104 (unmaintained third-party components) — the 2024 compromise was widely publicized; lack of remediation over two years represents a governance failure addressable through NIST AC-20 (Use of External Systems) policy enforcement.

NIST Phase: Post Incident

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

Controls: CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), NIST AC-20 (Use Of

External Systems), NIST AC-4 (Information Flow Enforcement)

Compensating: For teams without a commercial SCA (Software Composition Analysis) tool: implement a pre-commit Git hook using a simple bash script that greps all staged HTML/JS/template files for externally hosted script src attributes and fails the commit if any non-allowlisted CDN domain is detected. For CSP deployment without a WAF: add the Content-Security-Policy response header directly in Nginx (`add_header Content-Security-Policy "script-src 'self' [allowlisted-origins];"`) or Apache (`Header set Content-Security-Policy`). Deploy CSP in report-only mode first (`Content-Security-Policy-Report-Only`) directing violations to a free report endpoint (e.g., report-uri.com free tier) to identify remaining unauthorized script loads before enforcing — this prevents a broken production site while building the allowlist.

Evidence: As the post-incident documentation package, compile: (1) the complete inventory of all third-party script dependencies discovered during the audit, with CDN owner, acquisition history where known, and SRI status; (2) the HAR files and proxy logs captured during Steps 1–2 documenting the Funnull CDN HTTP 401 credential-prompt mechanism; (3) the git diff patch from Step 3 showing all eradication changes; and (4) the 48–72 hour clean DNS/proxy logs from Step 4. This package constitutes the lessons-learned evidence base per NIST 800-61r3 §4 and should be retained for any regulatory inquiry given that credential harvesting from consumer-facing sites (Toshiba, Muji, Samsung) may implicate user PII under applicable privacy regulations.

Detection Guidance

Primary detection method: scan all production HTML, JavaScript bundles, CMS templates, and tag manager configurations for any reference to polyfill[.]io (including cdn.polyfill.io, www.polyfill.io, and any subdomain). In proxy or DNS logs, query for resolved requests to polyfill[.]io from production web servers or CI/CD pipelines. In SIEM, alert on HTTP 401 responses where the host header or referrer contains polyfill[.]io. Browser-side telemetry (if available via RUM or CSP reporting): look for script load failures or authentication challenge events tied to polyfill[.]io origins. IOCs: domain polyfill[.]io (treat all subdomains as compromised); any script src attribute value containing 'polyfill.io'. Confidence: high for domain IOC (confirmed active and serving malicious 401 responses per BleepingComputer reporting, May 2026). File hashes and IP ranges are not available in current reporting; focus on domain-based IOCs and script tag scanning. For ongoing monitoring, reference NIST SI-4 (Information System Monitoring) applied to external dependency endpoints, and CIS 2.1 extended to web application source file monitoring for unauthorized script tag modifications.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`polyfill.io`	Reactivated CDN domain serving HTTP 401 responses to trigger browser credential prompts on dependent sites; operated by Funnull CDN post-2024 acquisition	HIGH
DOMAIN	`cdn.polyfill.io`	Primary subdomain used for script delivery from polyfill[.]io; treat all subdomains as compromised	HIGH

Framework Mappings

MITRE-ATTACK

- **T1556** — Modify Authentication Process
- **T1056.003** — Web Portal Capture
- **T1059.007** — JavaScript
- **T1189** — Drive-by Compromise
- **T1566.002** — Spearphishing Link
- **T1584.001** — Domains
- **T1071.001** — Web Protocols
- **T1195.002** — Compromise Software Supply Chain

NIST-800-53R5

- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **CM-3** — Configuration Change Control
- **SA-4** — Acquisition Process
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A06:2021** — Vulnerable and Outdated Components

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **16.4** — Establish and Manage an Inventory of Third-Party Software Components
- **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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1556	Modify Authentication Process	Credential-Access
T1056.003	Web Portal Capture	Collection
T1059.007	JavaScript	Execution
T1189	Drive-by Compromise	Initial-Access
T1566.002	Spearphishing Link	Initial-Access
T1584.001	Domains	Resource-Development
T1071.001	Web Protocols	Command-And-Control
T1195.002	Compromise Software Supply Chain	Initial-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/suspicious-polyfill-...	T3
Suspicious Polyfill login prompts pop up on Toshiba, Muji ...	https://www.bleepingcomputer.com/news/security/suspicious-polyfill-...	T3
Polyfill.io Supply Chain Attack: What You Need to Know	https://blog.qualys.com/vulnerabilities-threat-research/2024/06/28/...	T3
Polyfill.io Supply Chain Attack Explained	https://www.sonatype.com/blog/polyfill.io-supply-chain-attack-hits-...	T3
Polyfill Supply Chain Attack: Details and Fixes | FOSSA Blog	https://fossa.com/blog/polyfill-supply-chain-attack-details-fixes/	T3

Generated 2026-06-06 14:05 UTC by TJS Security Command Center