INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-15 13:50 UTC

Steganography-Laced JPEG Lure Campaign Abuses Cloudflare and WeTransfer to Stage Multi-Layer Malware Delivery

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0466
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Windows endpoints (PowerShell, WMI), Microsoft .NET Task Scheduler open-source library, Cloudflare Workers, Cloudflare R2 storage
Published	2026-06-15T03:16:00
Discovery Source	Rss

Executive Summary

An active campaign is delivering persistent malware to Windows endpoints by hiding payloads inside JPEG image files and staging them through Cloudflare's trusted infrastructure, bypassing most network filtering controls. The attack chain begins with a link sent via WeTransfer, moves through obfuscated PowerShell and WMI execution, and ends with a trojanized .NET scheduling library that gives attackers durable footholds on compromised systems. Organizations whose employees receive external file-sharing links and whose security controls rely on URL reputation or file-type inspection face the highest exposure.

Technical Analysis

This multi-stage campaign (no CVE assigned) targets Windows endpoints through a JavaScript dropper distributed via WeTransfer. The dropper executes ROT13-obfuscated PowerShell via WMI (T1047, T1059.001, T1140) to evade endpoint monitoring tools that inspect process trees rather than WMI-spawned execution. Subsequent stages retrieve JPEG files from Cloudflare Workers and Cloudflare R2 storage; those images contain Base64-encoded payloads embedded using modified delimiters, a steganographic technique (T1027, T1027.003) designed to defeat file-type inspection and signature-based detection. The final payload is a trojanized version of an open-source .NET Task Scheduler library (T1053.005), establishing persistence via scheduled tasks. Relevant weaknesses: CWE-116 (Improper Encoding or Escaping of Output), CWE-506 (Embedded Malicious Code), CWE-693 (Protection Mechanism Failure). No patch is available; mitigation is detection- and policy-based. The SANS ISC diary identifies this as a recurring technique with growing adversary

adoption.

Action Checklist

1. Step 1: Containment, Block WeTransfer (wetransfer.com) at the web proxy or DNS layer for managed endpoints where business need does not require it. Implement egress rules to alert on or block outbound connections to Cloudflare Workers subdomains (*.workers.dev) and Cloudflare R2 bucket URLs (*.r2.cloudflarestorage.com) that are not explicitly approved. Review firewall and proxy logs for connections to these destinations within the past 30 days.
2. Step 2: Detection, Query endpoint logs for WMI-spawned PowerShell execution: Windows Event ID 4688 (process creation) where parent process is WmiPrvSE.exe and child is powershell.exe or cmd.exe. Search for PowerShell command-line arguments containing ROT13 artifacts or Base64 strings with non-standard delimiters. In SIEM, correlate WeTransfer download events with subsequent WMI or PowerShell activity within the same user session. Check scheduled tasks created or modified in the past 30 days for tasks invoking unusual .NET DLLs (NIST AU-6, CIS 8.2).
3. Step 3: Eradication, On confirmed-compromised hosts, enumerate all scheduled tasks and remove tasks pointing to unrecognized or unsigned .NET assemblies. Identify and remove the trojanized Task Scheduler DLL; validate DLL integrity against known-good hashes of the legitimate open-source library. Revoke and rotate credentials for any accounts that executed suspicious PowerShell sessions (D3-CRO). Re-image endpoints where full scope of execution cannot be determined.
4. Step 4: Recovery, After re-imaging or DLL removal, verify no residual scheduled tasks reference the malicious library using 'schtasks /query /fo LIST /v' and cross-check against an approved task baseline. Monitor for recurrence of WMI-spawned PowerShell for at least 14 days post-remediation. Validate that egress controls blocking the identified Cloudflare delivery infrastructure remain in place and are logging (NIST AU-12, CIS 8.2).
5. Step 5: Post-Incident, This campaign exposed gaps in file-type inspection (JPEG content not scanned for embedded payloads), over-reliance on URL reputation controls that implicitly trust Cloudflare infrastructure, and insufficient WMI execution monitoring. Evaluate whether your EDR policy covers WMI-spawned child processes. Implement or tune detection rules for WMI process creation events (T1047). Review whether outbound connections to cloud storage and serverless platforms require additional inspection controls (NIST AC-4, behavioral monitoring equivalent to NIST SI-4). Refer to the SOC Prime coverage for community detection rules aligned to this campaign.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to senior IR leadership and legal/compliance if any compromised endpoint is confirmed to have accessed, processed, or stored PII, PHI, or financial data during the period of trojanized DLL persistence, as multi-jurisdictional breach notification obligations (GDPR 72-hour, HIPAA 60-day, SEC 4-day for material incidents) may be triggered; also escalate if the WMI/PowerShell execution chain is confirmed active on more than five endpoints, indicating lateral movement beyond initial access.

Recovery Notes	After DLL removal or reimage, validate the integrity of the legitimate open-source .NET Task Scheduler library against its published release hash before restoring scheduled task functionality on any affected endpoint. Monitor all recovered endpoints for WMI-spawned process creation and outbound connections to Cloudflare Workers or R2 subdomains for a minimum of 14 days, as the multi-layer delivery chain may have staged secondary implants beyond the identified Task Scheduler DLL. Confirm that file inspection controls at the email gateway and web proxy have been updated to perform content-based analysis of JPEG files (not solely relying on MIME type or URL reputation) before lifting any WeTransfer or Cloudflare egress restrictions imposed during containment.
Forensic Artifacts	Lure JPEG file retained in user Downloads or %TEMP% directory: contains the steganographically embedded payload extractable via binwalk or manual LSB analysis; primary evidence of the initial delivery vector and payload staging mechanism specific to this campaign. | Windows PowerShell ScriptBlock Log (Event ID 4104) entries for the suspect user session: will contain the decoded ROT13 or Base64 obfuscated stager that retrieved the secondary payload from Cloudflare R2 or Workers infrastructure, capturing the exact de-obfuscated command sequence unique to this multi-layer chain. | Trojanized .NET Task Scheduler DLL on disk (path likely under %APPDATA%, %PROGRAMDATA%, or a user-writable directory): the campaign's persistence mechanism; SHA-256 hash comparison against the legitimate open-source library release will confirm trojanization, and strings analysis will surface embedded C2 callback logic. | Windows Scheduled Task XML definitions at C:\Windows\System32\Tasks\ and corresponding registry entries at HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks: records the attacker-created persistence task invoking the malicious DLL, including creation timestamp, user account context, and trigger configuration. | Proxy and DNS query logs covering the 30-day pre-containment window: specifically HTTP GET requests to wetransfer.com (lure delivery), subsequent DNS resolutions and HTTPS connections to *.workers.dev (payload orchestration layer), and HTTPS connections to *.r2.cloudflarestorage.com (binary staging), reconstructing the full three-tier delivery chain unique to this campaign's abuse of Cloudflare's trusted infrastructure.

Per-Action IR Details

Step 1: Containment — Block WeTransfer (wetransfer.com) at the web proxy or DNS layer for managed endpoints where business need does not require it. Implement egress rules to alert on or block outbound connections to Cloudflare Workers subdomains (*.workers.dev) and Cloudflare R2 bucket URLs (*.r2.cloudflarestorage.com) that are not explicitly approved. Review firewall and proxy logs for connections to these destinations within the past 30 days.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: For teams without a managed proxy, deploy Windows Firewall GPO rules or host-based iptables to block outbound DNS resolution and HTTPS connections to *.workers.dev and *.r2.cloudflarestorage.com. Use Pi-hole or a local DNS sinkhole (bind9 with RPZ) to block wetransfer.com and the Cloudflare delivery subdomains fleet-wide at minimal cost. Capture current proxy/DNS logs before applying blocks to preserve evidence of prior callbacks.

Evidence: Before applying egress blocks, export the past 30 days of proxy or DNS logs and capture any live outbound TCP sessions to *.workers.dev or *.r2.cloudflarestorage.com using 'netstat -ano' or 'Get-NetTCPConnection | Where-Object {$_.State -eq "Established"}' on suspect endpoints. Preserve full HTTP request/response logs including URI paths and User-Agent strings for WeTransfer download events, as the lure JPEG filename and referral chain are required to reconstruct the initial delivery vector.

Step 2: Detection — Query endpoint logs for WMI-spawned PowerShell execution: Windows Event ID 4688 (process creation) where parent process is WmiPrvSE.exe and child is powershell.exe or cmd.exe. Search for PowerShell command-line arguments containing ROT13 artifacts or Base64 strings with non-standard delimiters. In SIEM, correlate WeTransfer download events with subsequent WMI or PowerShell activity within the same user session. Check scheduled tasks created or modified in the past 30 days for tasks invoking unusual .NET DLLs (NIST AU-6, CIS 8.2).

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), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, deploy Sysmon with SwiftOnSecurity's config (minimum: ProcessCreate event ID 1 with full command-line logging, and WMI activity event IDs 19/20/21). Run the following on each endpoint to surface WMI-spawned PowerShell: 'Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4688} | Where-Object {$_.Message -match "WmiPrvSE" -and $_.Message -match "powershell"}'. For scheduled task inspection without SIEM, run 'schtasks /query /fo CSV /v > tasks_baseline.csv' and diff against a known-good export, filtering for tasks whose action path contains '.dll' or references AppData, Temp, or ProgramData directories. Use the free Sigma rule 'proc_creation_win_wmi_spwaned_powershell' converted to a PowerShell query for offline log grep.

Evidence: This step is analytic and does not alter live state, but before pivoting to any identified host for deeper investigation, first capture: (1) full PowerShell ScriptBlock Logging output from HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging (Event ID 4104) for the suspect user session, which will contain the decoded ROT13 or Base64 stager payload; (2) the Windows Scheduled Task XML definitions from C:\Windows\System32\Tasks\ for any tasks created or modified in the campaign window; (3) WMI repository artifacts at C:\Windows\System32\wbem\Repository\ which may contain persisted WMI subscriptions used as a secondary persistence mechanism alongside the .NET Task Scheduler DLL.

Step 3: Eradication — On confirmed-compromised hosts, enumerate all scheduled tasks and remove tasks pointing to unrecognized or unsigned .NET assemblies. Identify and remove the trojanized Task Scheduler DLL; validate DLL integrity against known-good hashes of the legitimate open-source library. Revoke and rotate credentials for any accounts that executed suspicious PowerShell sessions (D3-CRO). Re-image endpoints where full scope of execution cannot be determined.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Without EDR, use Sysinternals Autoruns (free) to enumerate all scheduled task entries and flag any task whose image path is not digitally signed by a trusted publisher — specifically look for .NET DLL paths under AppData, ProgramData, or user-writable directories. To validate DLL integrity, compute SHA-256 of the suspect DLL with 'Get-FileHash -Algorithm SHA256 ' and compare against the published hash of the legitimate open-source .NET Task Scheduler library from its official GitHub release page. For credential rotation, prioritize any account whose LSASS process was accessible from the WmiPrvSE.exe execution context, as the multi-layer PowerShell chain may have invoked Invoke-Mimikatz or similar memory-resident credential access tooling.

Evidence: CRITICAL — order of volatility must be observed before any eradication action on a live host. Capture before revoking credentials, removing DLLs, or reimaging: (1) full RAM acquisition using WinPmem or Magnet RAM Capture to preserve in-memory remnants of the steganography decoder routine and any injected shellcode; (2) 'Get-NetTCPConnection -State Established' and 'netstat -ano' to record active C2 connections at the time of eradication; (3) full copy of the trojanized DLL and any co-located files before deletion, preserved with SHA-256 hash for IOC sharing; (4) export of HKCU and HKLM Run/RunOnce registry keys and the full scheduled task XML corpus from C:\Windows\System32\Tasks\; (5) copy of the lure JPEG file if still present in the user's Downloads or Temp directory, as it contains the embedded payload and constitutes primary forensic evidence of the delivery mechanism.

Step 4: Recovery — After re-imaging or DLL removal, verify no residual scheduled tasks reference the malicious library using 'schtasks /query /fo LIST /v' and cross-check against an approved task baseline. Monitor for recurrence of WMI-spawned PowerShell for at least 14 days post-remediation. Validate that egress controls blocking the identified Cloudflare delivery infrastructure remain in place and are logging (NIST AU-12, CIS 8.2).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-12 (Audit Record Generation), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Without a SIEM for continuous 14-day monitoring, schedule a daily cron-equivalent task using Windows Task Scheduler to run 'Get-WinEvent -LogName Security -FilterXPath' against Event ID 4688 filtered for WmiPrvSE.exe parent processes and output results to a dated log file for human review each morning. Use osquery with the scheduled_tasks table ('SELECT * FROM scheduled_tasks WHERE action LIKE "%.dll%";') run as a daily query to catch re-emergence of DLL-invoking tasks. Confirm egress firewall rules are still active post-reimage by running 'netsh advfirewall firewall show rule name=all' and verifying the *.workers.dev and *.r2.cloudflarestorage.com block rules are present.

Evidence: Before returning a remediated endpoint to production, verify system integrity by comparing the SHA-256 hash of the reinstalled or validated .NET Task Scheduler DLL against the known-good hash from the official open-source library release. Confirm the Windows Scheduled Task XML store at C:\Windows\System32\Tasks\ contains only baseline-approved entries. If the host was reimaged rather than surgically cleaned, verify the image source itself has not been compromised by checking the image hash against the golden image repository record.

Step 5: Post-Incident — This campaign exposed gaps in file-type inspection (JPEG content not scanned for embedded payloads), over-reliance on URL reputation controls that implicitly trust Cloudflare infrastructure, and insufficient WMI execution monitoring. Evaluate whether your EDR policy covers WMI-spawned child processes. Implement or tune detection rules for WMI process creation events (T1047). Review whether outbound connections to cloud storage and serverless platforms require additional inspection controls (NIST AC-4, NIST SI-4 equivalent behavioral monitoring). Refer to the SOC Prime coverage for community detection rules aligned to this campaign.

NIST Phase: Post Incident

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For teams without EDR, implement YARA rules targeting the steganography pattern characteristic of this campaign (sequential LSB extraction from JPEG EXIF or pixel data preceding a PE header magic byte sequence) and run them via ClamAV custom rule sets on email gateway and file share ingestion points. Deploy the Sigma rule 'win_wmi_spawned_powershell' (available on the SigmaHQ GitHub repository) converted to a Windows Event Log PowerShell query for standing detection. Conduct a manual review of all externally received JPEG files in the past 60 days using 'binwalk -e .jpg' to identify files with embedded executable content or anomalous appended data segments.

Evidence: Post-incident evidence preservation for lessons learned should include: the full decoded PowerShell stager payload recovered from ScriptBlock logs (Event ID 4104), the extracted embedded binary from the lure JPEG (recovered via steganography analysis), all proxy/DNS log exports showing the WeTransfer-to-Cloudflare-Workers delivery chain, and the trojanized .NET Task Scheduler DLL binary with hash — retained in a password-protected evidence archive for the organization's defined retention period per NIST AU-11 (Audit Record Retention) policy to support potential regulatory reporting or law enforcement referral.

Detection Guidance

Primary detection surface is endpoint telemetry. Alert on Windows Event ID 4688 where ParentProcessName contains 'WmiPrvSE.exe' and NewProcessName contains 'powershell.exe' or 'cmd.exe'. Supplement with PowerShell Script Block Logging (Event ID 4104); look for Base64-encoded strings, ROT13 character patterns (letter-shifted strings), or Invoke-Expression with decoded content. At the network layer, flag HTTP GET requests to *.workers.dev and *.r2.cloudflarestorage.com that return image/jpeg content-type but are followed immediately by PowerShell or .NET execution on the requesting host. In email and file-sharing gateway logs, correlate WeTransfer download events with subsequent suspicious process chains on the same host within a short time window. For scheduled task persistence, query the registry path HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks and HKLM\...\Tree for tasks created after the suspected compromise window. Behavioral indicators: JPEG files retrieved over HTTPS that trigger downstream process execution; PowerShell commands that perform string replacement operations on retrieved content before execution; new scheduled tasks invoking DLLs from user-writable directories. D3FEND countermeasures applicable: D3-SFA (System File Analysis) for DLL and scheduled task monitoring, D3-UAP (User Account Permissions) to restrict task creation rights, D3-MFA (Multi-factor Authentication) to limit credential abuse post-compromise.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`*.workers.dev`	Cloudflare Workers subdomain pattern used to host and serve steganographic JPEG payloads in this campaign	MEDIUM
DOMAIN	`*.r2.cloudflarestorage.com`	Cloudflare R2 object storage pattern used for payload staging	MEDIUM
DOMAIN	`wetransfer.com`	Initial access vector — JavaScript dropper delivered via WeTransfer link	MEDIUM
URL	`https://isc.sans.edu/diary/The%20Evil%20MSI%20Background%20is%20Back!/33054`	SANS ISC diary documenting this campaign — primary source reference	HIGH

Framework Mappings

MITRE-ATTACK

- **T1059.001** — PowerShell
- **T1027** — Obfuscated Files or Information
- **T1027.003** — Steganography
- **T1105** — Ingress Tool Transfer
- **T1218** — System Binary Proxy Execution
- **T1566.002** — Spearphishing Link
- **T1047** — Windows Management Instrumentation
- **T1140** — Deobfuscate/Decode Files or Information

- **T1102** — Web Service
- **T1566** — Phishing
- **T1053.005** — Scheduled Task

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection

ISO-27001-2022

- **A.5.34** — Privacy and protection of personal information
- **A.5.23** — Information security for use of cloud services

CIS-V8

- **8.2** — Collect Audit Logs

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1059.001	PowerShell	Execution
T1027	Obfuscated Files or Information	Defense-Evasion
T1027.003	Steganography	Defense-Evasion
T1105	Ingress Tool Transfer	Command-And-Control
T1218	System Binary Proxy Execution	Defense-Evasion
T1566.002	Spearphishing Link	Initial-Access
T1047	Windows Management Instrumentation	Execution
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1102	Web Service	Command-And-Control
T1566	Phishing	Initial-Access

Technique ID	Technique Name	Tactic
T1053.005	Scheduled Task	Execution

Sources

Source	URL	Tier
Security News	https://isc.sans.edu/diary/The%20Evil%20MSI%20Background%20is%20Bac...	T1
The Evil MSI Background Returns via Cloudflare and WMI	https://socprime.com/active-threats/the-evil-msi-background-returns/	T3
Using Cloudflare Workers with C# - Cloudflare Community	https://community.cloudflare.com/t/using-cloudflare-workers-with-c/...	T3
Support for Cloudflare Workers runtime · Issue #2637 · mswjs/msw	https://github.com/mswjs/msw/issues/2637	T3
Potential Security Concerns with Workers & Pages	https://community.cloudflare.com/t/potential-security-concerns-with...	T3

Generated 2026-06-15 13:50 UTC by TJS Security Command Center