INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-17 07:13 UTC

ClickFix Loader Ecosystem Expands: BabaDeda, Lorem Ipsum, and Potemkin Loaders Targeting Education, Finance, and Enterprise

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0493
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Windows, macOS, Microsoft Edge, Chromium-based browsers, WordPress (compromised sites), Microsoft Defender, Node.js (outdated v7.10.1)
Published	2026-06-16T13:41:28
Discovery Source	Rss

Executive Summary

Three purpose-built malware loader families, BabaDeda, Lorem Ipsum, and Potemkin, are being deployed through ClickFix social engineering campaigns that trick users into executing malicious commands via fake browser errors or CAPTCHA dialogs. The Lorem Ipsum chain is attributed with high confidence to Vanilla Tempest, a ransomware operator linked to Rhysida deployments; the Potemkin chain achieved lateral movement across 11 or more hosts and reached domain controllers in at least one confirmed incident. Organizations in education, finance, and enterprise sectors running Windows, macOS, and Chromium-based browsers are at elevated risk of ransomware deployment, credential theft, and network-wide compromise.

Technical Analysis

ClickFix is a social engineering technique that presents fake browser error prompts or CAPTCHA-style dialogs, directing users to execute PowerShell commands or clipboard-injected payloads. Three distinct loader chains have been documented: BabaDeda, Lorem Ipsum (attributed to Vanilla Tempest/Rhysida with high confidence), and Potemkin. Each loader is purpose-built for staged payload delivery and defense evasion, departing from commodity tooling. Delivery infrastructure includes compromised WordPress sites and outdated Node.js v7.10.1, indicating deliberate targeting of unpatched environments. The Potemkin chain demonstrated lateral movement across 11 or more hosts with domain controller access. CWE coverage: CWE-94 (code injection via PowerShell execution), CWE-427 (uncontrolled search path enabling DLL hijacking via T1574.002), CWE-506

(embedded malicious code within loaders). MITRE ATT&CK techniques span initial access (T1566 phishing, T1204.002 malicious file execution), execution (T1059.001 PowerShell), persistence (T1547.001 registry run keys, T1543 service creation), defense evasion (T1562.001 security tool impairment, T1027 obfuscation, T1218.005 mshta), credential access (T1539 session cookie theft, T1056 input capture), lateral movement (T1021.002 SMB, T1021.006 WinRM, T1021 remote services), collection (T1113 screen capture), exfiltration (T1041), and impact (T1486 ransomware). No CVE is assigned; no vendor patch applies, this is a technique-based campaign requiring behavioral controls.

Action Checklist

1. Step 1: Containment. Immediately audit PowerShell execution policy across all endpoints; block execution of unsigned PowerShell scripts. Isolate any Windows host exhibiting unexpected mshta.exe (T1218.005), wscript.exe, or node.exe process invocations. Segment or quarantine hosts that contacted known WordPress-hosted delivery domains. Apply NIST AC-4 (Information Flow Enforcement, block unsigned external connections) to restrict outbound connections from endpoints to unclassified external destinations.
2. Step 2: Detection. Query EDR/SIEM for: (1) PowerShell spawned from browser processes (chrome.exe, msedge.exe, MicrosoftEdge.exe); (2) clipboard-read operations followed within 30 seconds by cmd.exe or powershell.exe execution; (3) mshta.exe or wscript.exe child processes of user-space applications; (4) node.exe executing on any host; (5) outbound HTTP/HTTPS to compromised WordPress infrastructure (look for POST to /wp-content/ or /wp-admin/ paths originating from workstations/endpoints, not legitimate server-to-server traffic); (6) T1482 domain trust discovery commands (nltest, dsquery); (7) T1574.002 DLL side-loading indicators, unsigned DLLs loaded from user-writable paths. Enable AU-2 (Event Logging) covering process creation (Event ID 4688), PowerShell script block logging (Event ID 4104), and WinRM activity (Event ID 4103). Map to D3-LAM (Local Account Monitoring) and D3-SFA (System File Analysis).
3. Step 3: Eradication. (1) If Node.js is deployed in your environment, audit for outdated versions and replace with a current LTS release per vendor guidance; no supported upgrade path exists for v7.x. (2) Remove or disable mshta.exe via AppLocker or Windows Defender Application Control where not operationally required. (3) If you host WordPress sites, audit them for compromise indicators: unexpected admin accounts, modified theme/plugin files, unauthorized file uploads. (4) Rotate credentials for any account that authenticated from a host with confirmed loader activity, per D3-CRO (Credential Rotation). (5) Apply CIS 7.3 (Automated OS Patch Management) and CIS 7.4 (Automated Application Patch Management) to close the unpatched-environment targeting vector.
4. Step 4: Recovery. (1) Validate that no persistence mechanisms remain: check registry run keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run), scheduled tasks, and installed services for entries created during the suspected compromise window per T1547.001 and T1543 indicators. (2) Confirm Microsoft Defender or equivalent EDR signatures are current and real-time protection is active; Potemkin was observed attempting security tool impairment (T1562.001). (3) Monitor domain controllers for anomalous authentication events (Event IDs 4624, 4625, 4672, 4768) for a minimum of 14 days post-remediation. (4) Verify DLL load order on affected hosts, re-image if DLL hijacking via T1574.002 cannot be ruled out. (5) Validate audit logging continuity per AU-9 (Protection of Audit Information) to confirm logs were not tampered with during the incident.
5. Step 5: Post-Incident. (1) This campaign exposed a user-execution gap: implement security awareness training specifically covering ClickFix-style prompts. Users should never execute commands pasted from a browser dialog or CAPTCHA. Map to CIS 7.1 (Vulnerability Management Process). (2) Enforce CIS 6.3

(MFA for Externally-Exposed Applications) and CIS 6.4 (MFA for Remote Network Access); T1078 valid account abuse and T1021 lateral movement are reduced by MFA on all remote access paths. (3) Apply AC-6 (Least Privilege); the Potemkin chain's domain controller reach indicates excessive privilege on compromised accounts. (4) Establish or update a ClickFix-specific playbook referencing T1204.002 and T1059.001 detection logic. (5) Review third-party WordPress and Node.js assets in your supply chain per CIS 2.2 (Ensure Authorized Software is Currently Supported); outdated infrastructure was deliberately targeted.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to senior IR leadership, legal counsel, and executive stakeholders if: any host with confirmed loader activity (BabaDeda, Lorem Ipsum, or Potemkin) is found to have authenticated to a domain controller (Event ID 4768/4769 from workstation IPs), indicating Potemkin-style lateral movement with Vanilla Tempest/Rhysida ransomware pre-deployment risk; or if data staging, exfiltration indicators, or Rhysida ransomware binary artifacts are discovered, triggering breach notification assessment under applicable regulatory frameworks (HIPAA for healthcare, GLBA for finance, applicable state privacy statutes for education sector targets).
Recovery Notes	Given Vanilla Tempest's confirmed association with Rhysida ransomware deployments, recovery must not be declared complete until a full lateral movement scope is established — specifically, all hosts that received authenticated connections from the Potemkin-compromised chain must be individually validated, not just the initial patient-zero host. Domain controller integrity must be verified including AD replication metadata, GPO modifications, and new or modified admin accounts, as Rhysida pre-deployment activity commonly includes AD persistence before ransomware execution. Maintain elevated monitoring on domain controllers and privileged accounts (Security Event IDs 4624 type 3, 4672, 4768) for a minimum of 30 days post-eradication given the confirmed 11-or-more-host lateral movement scope observed in at least one deployment of this campaign.

Forensic Artifacts	Clipboard history artifacts: Windows 10/11 Clipboard History (enabled via Win+V) stored in %LOCALAPPDATA%\Microsoft\Windows\Clipboard — ClickFix attack deposits the malicious PowerShell or cmd.exe command into clipboard for user-pasted execution; this is the primary evidence of the social engineering delivery mechanism and may contain the exact malicious command string used. | PowerShell Script Block logs (Event ID 4104, Microsoft-Windows-PowerShell/Operational): Contains decoded content of all PowerShell script blocks executed, including ClickFix-delivered IEX payloads with Base64-encoded or obfuscated BabaDeda/Lorem Ipsum/Potemkin loader stager commands — critical for reconstructing the full execution chain from initial user paste through loader staging. | Node.js v7.10.1 working directory and process artifacts: %APPDATA%, %TEMP%, and the directory from which node.exe was invoked — Lorem Ipsum and Potemkin chains use outdated Node.js as a runtime for loader execution; npm cache artifacts, loaded .js files, and child process invocations from node.exe are specific to this campaign's execution methodology. | WordPress server-side artifacts on compromised delivery infrastructure: wp-content/uploads/ directory for unauthorized file uploads (staged loader payloads), wp_users database table for unauthorized admin account creation, and web server access logs filtered for POST requests from victim endpoints to /wp-admin/ or /wp-content/ paths — these establish the delivery infrastructure tier of the ClickFix campaign chain. | Windows Security Event IDs 4768/4769/4672 on domain controllers correlated with workstation source IPs: Specific to the Potemkin lateral movement chain's confirmed domain controller reach — Kerberos TGT and service ticket requests from workstation-class IPs to DCs, combined with special privilege assignment events, reconstruct the privilege escalation and lateral movement path used by the Vanilla Tempest operator in documented intrusions of this campaign.

Per-Action IR Details

Step 1: Containment — Immediately audit PowerShell execution policy across all endpoints; block execution of unsigned PowerShell scripts. Isolate any Windows host exhibiting unexpected mshta.exe (T1218.005), wscript.exe, or node.exe (v7.10.1) process invocations. Segment or quarantine hosts that contacted known WordPress-hosted delivery domains. Apply NIST AC-4 (Information Flow Enforcement) to restrict outbound connections from endpoints to unclassified external destinations.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST AC-3 (Access Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Use PowerShell Group Policy (Set-ExecutionPolicy AllSigned) pushed via GPO or local script on each host. Block outbound TCP 80/443 from endpoints to unknown external IPs using Windows Firewall netsh rules: 'netsh advfirewall firewall add rule name="Block Unsigned PS" dir=out action=block program="%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe"' with scope restricted to known-good proxy/egress IPs. Use Sysmon Event ID 1 (process creation) to identify mshta.exe and node.exe v7.10.1 invocations in lieu of EDR isolation — terminate via 'taskkill /F /IM mshta.exe' and unplug network cable or disable NIC via 'Disable-NetAdapter' on confirmed hosts.

Evidence: Acquire volatile evidence BEFORE isolating any host: (1) Full RAM image using WinPmem or DumpIt — ClickFix loaders may reside only in memory prior to writing staged payloads to disk; (2) Run 'Get-NetTCPConnection | Where-Object {$_.State -eq "Established"}' and 'netstat -ano' to capture active C2 or WordPress delivery connections; (3) Export running process list with parent PIDs via 'Get-WmiObject Win32_Process | Select-Object ProcessId, ParentProcessId, Name, CommandLine | Export-Csv'; (4) Capture clipboard contents via PowerShell 'Get-Clipboard' — ClickFix attacks stage malicious commands in the clipboard for user-pasted execution, and clipboard content is lost on reboot; (5) Collect Prefetch files from C:\Windows\Prefetch for mshta.exe, wscript.exe, node.exe, and powershell.exe before isolation wipes scheduled task or run-key–triggered prefetch updates.

Step 2: Detection — Query EDR/SIEM for: (1) PowerShell spawned from browser processes (chrome.exe, msedge.exe, MicrosoftEdge.exe); (2) clipboard-read operations followed within 30 seconds by cmd.exe or powershell.exe execution; (3) mshta.exe or wscript.exe child processes of user-space applications; (4) node.exe version 7.10.1 executing on any host; (5) outbound HTTP/HTTPS to compromised WordPress infrastructure (look for POST to /wp-content/ or /wp-admin/ paths from endpoints, not servers); (6) T1482 domain trust discovery commands (nltest, dsquery); (7) T1574.002 DLL side-loading indicators — unsigned DLLs loaded from user-writable paths. Enable AU-2 (Event Logging) covering process creation (Event ID 4688), PowerShell script block logging (Event ID 4104), and WinRM activity (Event ID 4103). Map to D3-LAM (Local Account Monitoring) and D3-SFA (System File Analysis).

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-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon with SwiftOnSecurity config (minimum: Event ID 1 process creation, Event ID 3 network connections, Event ID 7 image/DLL load, Event ID 11 file creation). Query collected Sysmon logs via PowerShell: 'Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" | Where-Object {$_.Message -like "*chrome.exe*" -and $_.Id -eq 1}' filtered for child processes powershell.exe or cmd.exe. Use Sigma rule 'proc_creation_win_browser_spawn_powershell' (available in SigmaHQ repository) converted to Windows Event Log XML queries. For node.exe v7.10.1 detection: query 'Get-WmiObject Win32_Process | Where-Object {$_.Name -eq "node.exe"}' and cross-check version via 'node --version' on discovered processes. Use Wireshark or tcpdump capture on egress points filtered for 'http.request.method == POST and http.request.uri contains "/wp-content/"'.

Evidence: This is a detection/analysis step that does not directly alter live state, but if triage confirms a live compromise during this phase, volatile capture (RAM, active connections, clipboard) must be completed before any containment action is taken. Key evidence sources for this campaign: (1) Windows Security Event Log — Event ID 4688 with 'ProcessCommandLine' enabled via Audit Policy, filtering ParentProcessName for chrome.exe, msedge.exe spawning powershell.exe or cmd.exe; (2) PowerShell Script Block Log (Event ID 4104 in Microsoft-Windows-PowerShell/Operational) — ClickFix payloads frequently invoke IEX (Invoke-Expression) or encoded -EncodedCommand strings; decode and extract Base64 payloads for IOC extraction; (3) DNS query logs or Sysmon Event ID 22 (DNS query) for resolution of compromised WordPress domains used as delivery infrastructure; (4) Sysmon Event ID 7 (ImageLoad) for unsigned DLLs loaded from %APPDATA%, %TEMP%, or other user-writable paths indicative of BabaDeda or Potemkin DLL side-loading; (5) Windows Security Event IDs 4768/4769 (Kerberos TGT/service ticket requests) from hosts exhibiting loader activity, as Potemkin achieved domain controller reach — anomalous ticket requests from workstations are a high-fidelity lateral movement indicator.

Step 3: Eradication — (1) Upgrade or remove Node.js v7.10.1 from all enterprise assets; no supported upgrade path exists for v7.x — replace with a current LTS release per vendor guidance. (2) Remove or disable mshta.exe via AppLocker or Windows Defender Application Control where not operationally required. (3) Audit and remediate any WordPress installations in your environment for compromise indicators: unexpected admin accounts, modified theme/plugin files, unauthorized file uploads. (4) Rotate credentials for any account that authenticated from a host with confirmed loader activity, per D3-CRO (Credential Rotation). (5) Apply CIS 7.3 (Automated OS Patch Management) and CIS 7.4 (Automated Application Patch Management) to close the unpatched-environment targeting vector.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: NIST AC-2 (Account Management), NIST SI-2 (Flaw Remediation) — [not in knowledge base; omitted per citation discipline], CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: For Node.js removal: run 'wmic product where "name like \'Node.js\'" get name,version' across hosts via PSExec or WMI remote query to enumerate v7.x instances; uninstall via 'msiexec /x {GUID} /quiet' or 'wmic product where name="Node.js" call uninstall'. For mshta.exe blocking without AppLocker: rename or ACL-restrict C:\Windows\System32\mshta.exe using 'icacls C:\Windows\System32\mshta.exe /deny Everyone:(X)'. For WordPress audit on a budget: run WPScan (free tier) against internal or externally hosted WordPress instances; manually diff wp-content/themes/ and wp-content/plugins/ directories against known-good checksums using 'Get-FileHash' in PowerShell. Rotate compromised credentials via Active Directory 'Set-ADAccountPassword' bulk script scoped to accounts with last logon from flagged hosts.

Evidence: Acquire volatile evidence BEFORE rotating credentials or uninstalling Node.js — credential rotation invalidates session tokens that may contain forensic attribution data, and uninstallation removes loader artifacts. Capture: (1) Node.js v7.10.1 process memory and working directory listing ('Get-Process node | Select-Object Id, Path, MainModule') — Lorem Ipsum and Potemkin chains have been observed staging secondary payloads from the Node.js runtime working directory; (2) Full export of WordPress database and file system from any WordPress instance under assessment BEFORE remediation — unexpected wp_users entries and modified wp_options (siteurl, home, active_plugins) are primary indicators of WordPress-layer compromise used as delivery infrastructure; (3) Windows Credential Manager vault dump ('cmdkey /list') and LSASS minidump (via ProcDump: 'procdump -ma lsass.exe lsass.dmp') from compromised hosts before credential rotation — Vanilla Tempest/Rhysida operators are known to harvest credentials for lateral movement prior to ransomware deployment; (4) Registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run before any remediation to preserve persistence evidence.

Step 4: Recovery — (1) Validate that no persistence mechanisms remain: check registry run keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run), scheduled tasks, and installed services for entries created during the suspected compromise window per T1547.001 and T1543 indicators. (2) Confirm Microsoft Defender or equivalent EDR signatures are current and real-time protection is active — Potemkin was observed attempting security tool impairment (T1562.001). (3) Monitor domain controllers for anomalous authentication events (Event IDs 4624, 4625, 4672, 4768) for a minimum of 14 days post-remediation. (4) Verify DLL load order on affected hosts — re-image if DLL hijacking via T1574.002 cannot be ruled out. (5) Validate audit logging continuity per AU-9 (Protection of Audit Information) to confirm logs were not tampered with during the incident.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-9 (Protection Of Audit Information), NIST AU-11 (Audit Record Retention), NIST AC-6 (Least Privilege), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 5.3 (Disable Dormant Accounts)

Compensating: Persistence validation without EDR: use Autoruns (Sysinternals) with VirusTotal integration enabled — run 'autorunsc.exe -a * -h -c -vt > autoruns_output.csv' and review entries with non-Microsoft signatures or unknown hashes. For scheduled task enumeration: 'schtasks /query /fo CSV /v > scheduled_tasks.csv' and filter for tasks created within the compromise window using creation timestamps. For Defender integrity check: 'Get-MpComputerStatus | Select-Object AMRunningMode, RealTimeProtectionEnabled, AntivirusSignatureLastUpdated' — Potemkin's security tool impairment attempts may have disabled real-time protection or altered exclusion lists ('Get-MpPreference | Select-Object ExclusionPath, ExclusionProcess'). Monitor DC authentication via free Windows Event Forwarding (WEF) to a central collector, filtering Security log for Event IDs 4624 (type 3 network logon), 4672 (special privileges), 4768 (TGT request) originating from previously compromised workstation IPs.

Evidence: Before reimaging any host for suspected DLL hijacking: (1) Collect Sysmon Event ID 7 (ImageLoad) logs showing the full DLL load sequence including unsigned DLL paths loaded by the vulnerable application — this establishes the specific DLL hijacking technique and affected binary for threat intel reporting; (2) Export the full registry hive (HKLM\SYSTEM, HKLM\SOFTWARE, HKCU) from affected hosts using 'reg save HKLM\SYSTEM C:\evidence\system.hiv' before reimaging — Potemkin persistence via services (T1543) and run keys (T1547.001) will be lost on reimage; (3) Capture Windows Event Log files (.evtx) from C:\Windows\System32\winevt\Logs\ — specifically Security.evtx, System.evtx, and Microsoft-Windows-PowerShell%4Operational.evtx — before reimaging,

as these contain the authentication and execution timeline needed for scope determination against the Potemkin lateral movement chain; (4) Collect Microsoft Defender exclusion list and quarantine history via 'Get-MpPreference' and 'Get-MpThreatDetection' to document what Potemkin's impairment attempts modified or suppressed.

Step 5: Post-Incident — (1) This campaign exposed a user-execution gap: implement security awareness training specifically covering ClickFix-style prompts — users should never execute commands pasted from a browser dialog or CAPTCHA. Map to CIS 7.1 (Vulnerability Management Process). (2) Enforce CIS 6.3 (MFA for Externally-Exposed Applications) and CIS 6.4 (MFA for Remote Network Access) — T1078 valid account abuse and T1021 lateral movement are reduced by MFA on all remote access paths. (3) Apply AC-6 (Least Privilege) — the Potemkin chain's domain controller reach indicates excessive privilege on compromised accounts. (4) Establish or update a ClickFix-specific playbook referencing T1204.002 and T1059.001 detection logic. (5) Review third-party WordPress and Node.js assets in your supply chain per CIS 2.2 (Ensure Authorized Software is Currently Supported) — outdated infrastructure was deliberately targeted.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.4 (Require MFA for Remote Network Access), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts)

Compensating: For ClickFix-specific user awareness with no LMS budget: distribute a one-page visual guide showing exactly what BabaDeda/Lorem Ipsum/Potemkin fake CAPTCHA and browser-error dialogs look like, with explicit instruction that no legitimate website requires users to open Run dialog or paste commands into PowerShell. For MFA without enterprise SSO: implement Windows Hello for Business (free, included in Windows 10/11) or deploy Duo MFA free tier (up to 10 users) on VPN and RDP gateways. For least-privilege remediation: run 'net localgroup administrators' on all workstations via PSExec sweep and remove any non-IT accounts; use 'Get-ADGroupMember "Domain Admins"' to audit domain admin membership and remove accounts not explicitly requiring DA rights — Potemkin's DC reach in this campaign is a direct indicator of over-privileged workstation accounts. Publish the ClickFix IOC set (WordPress delivery domains, mshta.exe invocation patterns, node.exe v7.10.1 hashes) to internal threat intel via a free MISP instance.

Evidence: Post-incident evidence collection for lessons-learned and regulatory review: (1) Reconstruct the full ClickFix social engineering chain — collect browser history and downloaded file artifacts from %USERPROFILE%\Downloads and %TEMP% on patient-zero host to document exactly which fake CAPTCHA or browser-error page initiated the clipboard-inject; (2) Compile a timeline of Vanilla Tempest / Rhysida-attributed TTPs observed in this incident against the known Lorem Ipsum kill chain — document which MITRE ATT&CK techniques were confirmed observed vs. not observed, to scope whether ransomware pre-deployment activity (data staging, exfiltration) occurred before containment; (3) Export Active Directory audit logs showing privilege escalation or lateral movement path used by the Potemkin chain to reach domain controllers — specifically Security Event ID 4672 (special privileges assigned) and 4776 (credential validation) from DC Security logs — to establish the privilege chain for post-incident remediation scoping; (4) Retain all collected .evtx, memory images, and WordPress forensic artifacts for a minimum period consistent with your organization's incident retention policy, as Vanilla Tempest/Rhysida incidents frequently trigger breach notification obligations under HIPAA, state privacy laws, or sector-specific regulations depending on data accessed.

Detection Guidance

Primary detection focus: PowerShell spawned from browser parent processes. Query for Event ID 4688 (process creation) where ParentImage contains chrome.exe, msedge.exe, or MicrosoftEdge.exe and NewProcessName contains powershell.exe or cmd.exe. Enable PowerShell Script Block Logging (Event ID

4104) and alert on base64-encoded payloads or IEX (Invoke-Expression) patterns in script blocks. Alert on mshta.exe (T1218.005) execution by any non-system parent process. Flag node.exe on any host, this specific version is a confirmed delivery infrastructure indicator. For lateral movement: monitor SMB authentication sequences (Event ID 4648) and WinRM session establishment (Event ID 91, 168) from workstation-to-workstation paths, which are anomalous in most environments. For T1539 session cookie theft: alert on browser process memory access by non-browser tools. For T1482 domain reconnaissance: alert on nltest /domain_trusts and dsquery executions from non-administrator accounts. D3-SFA (System File Analysis) applies to monitoring startup configuration changes. D3-LAM (Local Account Monitoring) applies to detecting new local accounts or privilege escalations created during post-compromise persistence. IOC enrichment: cross-reference outbound connections against compromised WordPress domains identified in threat intelligence feeds.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`compromised WordPress infrastructure (no specific domains confirmed in provided source data)`	WordPress sites used as delivery and staging infrastructure for ClickFix loader chains; specific domains not disclosed in available sources	LOW
URL	`/wp-content/ and /wp-admin/ POST endpoints on external domains`	Behavioral pattern — endpoints receiving POST requests from workstations (not servers) indicate use of compromised WordPress staging	MEDIUM
HASH	`not disclosed in available source data`	BabaDeda, Lorem Ipsum, and Potemkin loader hashes not provided in available sources; enrich from threat intelligence platform against campaign name	LOW
URL	`mshta.exe execution with remote URL argument`	mshta.exe leveraged via ClickFix social engineering campaigns to stage and execute remote malicious payloads for loader deployment targeting education, finance, and enterprise sectors.	HIGH
DOMAIN	`node.exe v7.10.1 network connections`	Outdated Node.js v7.10.1 used as staging infrastructure; any outbound connection from a node.exe process of this version warrants investigation	HIGH

Framework Mappings

MITRE-ATTACK

- **T1078** — Valid Accounts
- **T1486** — Data Encrypted for Impact
- **T1021.002** — SMB/Windows Admin Shares

- **T1113** — Screen Capture
- **T1055** — Process Injection
- **T1566** — Phishing
- **T1021.006** — Windows Remote Management
- **T1041** — Exfiltration Over C2 Channel
- **T1482** — Domain Trust Discovery
- **T1204.002** — Malicious File
- **T1090.003** — Multi-hop Proxy
- **T1574.002** — DLL Side-Loading
- **T1056** — Input Capture
- **T1102** — Web Service
- **T1218.005** — Mshta
- **T1059.001** — PowerShell
- **T1539** — Steal Web Session Cookie
- **T1021** — Remote Services
- **T1562.001** — Disable or Modify Tools
- **T1547** — Boot or Logon Autostart Execution
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1543** — Create or Modify System Process
- **T1583.001** — Domains
- **T1027** — Obfuscated Files or Information

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-17** — Remote Access
- **AC-3** — Access Enforcement

- **SI-10** — Information Input Validation
- **IR-4** — Incident Handling

OWASP-TOP10-2021

- **A03:2021** — Injection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **DE.CM-01** — Networks and network services are monitored

HIPAA-SECURITY

- **164.308(a)(7)(ii)(A)** — Data Backup Plan

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1078	Valid Accounts	Defense-Evasion
T1486	Data Encrypted for Impact	Impact
T1021.002	SMB/Windows Admin Shares	Lateral-Movement
T1113	Screen Capture	Collection
T1055	Process Injection	Defense-Evasion
T1566	Phishing	Initial-Access
T1021.006	Windows Remote Management	Lateral-Movement
T1041	Exfiltration Over C2 Channel	Exfiltration
T1482	Domain Trust Discovery	Discovery
T1204.002	Malicious File	Execution

Technique ID	Technique Name	Tactic
T1090.003	Multi-hop Proxy	Command-And-Control
T1574.002	DLL Side-Loading	Persistence
T1056	Input Capture	Collection
T1102	Web Service	Command-And-Control
T1218.005	Mshta	Defense-Evasion
T1059.001	PowerShell	Execution
T1539	Steal Web Session Cookie	Credential-Access
T1021	Remote Services	Lateral-Movement
T1562.001	Disable or Modify Tools	Defense-Evasion
T1547	Boot or Logon Autostart Execution	Persistence
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1543	Create or Modify System Process	Persistence
T1583.001	Domains	Resource-Development
T1027	Obfuscated Files or Information	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/clickfix-campaigns-expand-malware...	T3
Importance of Browser Updates and Browser Based Security Controls	https://techcommunity.microsoft.com/blog/coreinfrastructureandsecur...	T1
Critical Microsoft EDGE and Chromium browsers update for security	https://www.youtube.com/watch?v=XPjmLmW2xWU	T3
Critical Chromium Browser Vulnerability Under Active Exploit	https://isidefense.com/blog/critical-chromium-browser-vulnerability...	T3
Critical vulnerability with Google Chrome and Chromium based ...	https://ociso.ucla.edu/news/critical-vulnerability-google-chrome-an...	T1

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-17 07:13 UTC by TJS Security Command Center