INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-30 18:43 UTC

DeepLoad: Multi-Stage Loader Combining ClickFix, APC Injection, and WMI-Based Autonomous Reinfection for Credential Theft

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0125
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Windows systems (LockAppHost.exe, WMI subsystem, PowerShell, mshta.exe); Chrome and Firefox browsers; browser extensions; AnyDesk (used as lure filename)
Published	2026-03-30T11:47:00
Discovery Source	Rss

Executive Summary

DeepLoad is an active malware campaign documented by ReliaQuest that targets Windows systems through social engineering, then steals browser credentials from Chrome and Firefox. The threat's most significant business risk is a built-in WMI-based reinfection mechanism that reinstates the malware approximately three days after remediation without attacker interaction, meaning standard incident response procedures may produce a false sense of containment. Any organization with Windows endpoints, browser-stored credentials, and USB-connected devices is in scope; credential theft at scale can enable account takeover, privilege escalation, and downstream breaches.

Technical Analysis

DeepLoad is a multi-stage, fileless malware loader documented by ReliaQuest (March 2026) targeting Windows systems. No CVE is assigned; the campaign exploits user behavior, not a patched software vulnerability. Relevant CWEs: CWE-116 (improper encoding/escaping), CWE-506 (embedded malicious code), CWE-494 (download without integrity check), CWE-522 (insufficiently protected credentials), CWE-77 (command injection). Attack chain: (1) ClickFix social engineering lures users into manually executing malicious PowerShell commands, often via fake AnyDesk installer prompts. (2) AI-assisted obfuscation generates polymorphic PowerShell scripts per execution, undermining signature-based detection (MITRE T1027, T1027.010, T1059.001). (3) Payload is injected into LockAppHost.exe via Asynchronous Procedure Call (APC) injection (T1055.004), avoiding disk writes. (4) mshta.exe is used as a living-off-the-land binary for execution

(T1218.005). (5) WMI event subscriptions (T1546.003, T1047) establish autonomous reinfection approximately 72 hours post-cleanup without attacker interaction. (6) Primary payload harvests stored credentials from Chrome and Firefox (T1555.003, T1539); secondary capability spreads via USB (T1091). Browser extensions are also targeted (T1176). Attribution is unestablished. A separate actor claiming Kiss Loader affiliation and Malawi origin is unverified. No patch exists; mitigation is detection and configuration hardening.

Action Checklist

1. Containment: Immediately restrict PowerShell execution to signed scripts via Constrained Language Mode or AppLocker/WDAC policy on all Windows endpoints. Block mshta.exe execution via application control where operationally feasible. Isolate any host exhibiting LockAppHost.exe with unexpected child processes or network connections. Disable USB storage on endpoints not requiring it via Group Policy (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR, Start = 4).
2. Detection: Query WMI event subscriptions for unauthorized entries: run 'Get-WMIObject -Namespace root\subscription -Class __EventFilter', '__EventConsumer', and '__FilterToConsumerBinding' and alert on any entries not matching your baseline. Hunt for PowerShell processes spawned by user-interactive processes (explorer.exe, mshta.exe) with encoded or obfuscated command lines (Event ID 4104 with ScriptBlock logging enabled). Alert on LockAppHost.exe initiating network connections or spawning child processes. Review Windows Security Event ID 4688 for mshta.exe execution with unusual parent processes.
3. Eradication: Remove all unauthorized WMI event subscriptions identified in detection step using 'Remove-WMIObject' or WMI Explorer. Terminate and remediate any injected LockAppHost.exe processes. Force password rotation for all credentials stored in Chrome and Firefox on affected hosts, treat stored browser credentials as compromised. Re-image hosts where injection is confirmed rather than attempting in-place cleanup, given the fileless and reinfection characteristics. Verify WMI subscription baseline is clean before declaring eradication; recheck at 72-96 hours post-remediation given the documented reinfection delay.
4. Recovery: Validate WMI subscription baseline matches known-good state at 24, 48, and 96 hours post-remediation. Monitor for re-emergence of obfuscated PowerShell execution or LockAppHost.exe anomalies. Confirm browser credential stores have been cleared and users have rotated passwords for all accounts accessible from affected systems. Verify application control policies blocking mshta.exe and unsigned PowerShell remain enforced after any system updates.
5. Post-Incident: This campaign exposed three control gaps: (1) absence of PowerShell ScriptBlock logging and Constrained Language Mode, (2) no WMI subscription monitoring baseline, and (3) reliance on browser-stored credentials as a primary authentication store. Implement NIST SP 800-53 SI-3 (malicious code protection), SI-7 (software integrity), and AU-12 (audit record generation) controls. Evaluate credential manager or password manager deployment to reduce browser credential exposure. Add WMI persistence monitoring to SOC detection backlog as a standing hunt. Brief incident responders on the 72-hour reinfection window; close-out criteria must include WMI validation, not just process termination.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO and legal counsel immediately if browser credential stores on affected hosts contain credentials for financial systems, HR platforms, or any system processing PII or PHI, as credential exfiltration scope may trigger breach notification obligations under applicable regulations (GDPR 72-hour window, HIPAA, state breach laws); also escalate if WMI reinfection is confirmed post-remediation, as this indicates the organization lacks the detection capability to reliably close the incident without external IR support.
Recovery Notes	Recovery cannot be declared complete until WMI subscription state has been validated clean at the 96-hour mark post-remediation — the DeepLoad campaign's documented 72-hour autonomous reinfection window means any close-out before this checkpoint risks a false sense of containment. In parallel, treat all Chrome and Firefox credential stores on affected hosts as fully compromised regardless of whether exfiltration is confirmed, and require password rotation at the destination service (not just locally) for every account credential stored in those browsers. Maintain elevated WMI and PowerShell ScriptBlock log monitoring for 30 days post-incident to detect any secondary infections from credentials exfiltrated during the initial compromise.
Forensic Artifacts	WMI Repository binary files (%SystemRoot%\System32\wbem\Repository\OBJECTS.DATA and INDEX.BTR) — contain the DeepLoad reinfection subscription objects (__EventFilter, __EventConsumer, __FilterToConsumerBinding) that persist even if PowerShell-based queries return empty; required for confirming full eradication of the autonomous reinfection mechanism. | Process memory dump of LockAppHost.exe (captured via ProcDump before host isolation) — contains the APC-injected shellcode payload in-memory, which is the only copy of the fileless second-stage loader; critical for IOC extraction and sandbox behavioral analysis. | Chrome Login Data SQLite database (%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data) and Firefox logins.json plus key4.db (%APPDATA%\Mozilla\Firefox\Profiles*.default-release\) — confirm which credentials were stored and therefore in scope for exfiltration by DeepLoad's credential theft module. | PowerShell Operational Event Log (Microsoft-Windows-PowerShell%4Operational.evtx) filtered for Event ID 4104 (ScriptBlock Logging) — reconstructs the encoded/obfuscated command sequence used by the ClickFix-delivered loader stage, including any FromBase64String or IEX invocations that initiated mshta.exe and subsequent APC injection. | Windows Prefetch files for mshta.exe (%SystemRoot%\Prefetch\MSHTA.EXE-*.pf) and LockAppHost.exe — provide execution timestamps and loaded DLL lists that establish the attack timeline from ClickFix lure delivery through process injection, and confirm whether mshta.exe was invoked as a proxy for the initial loader stage.

Per-Action IR Details

Containment — Immediately restrict PowerShell execution to signed scripts via Constrained Language Mode or AppLocker/WDAC policy on all Windows endpoints. Block mshta.exe execution via application control where operationally feasible. Isolate any host exhibiting LockAppHost.exe with unexpected child processes or network connections. Disable USB storage on endpoints not requiring it via Group Policy (HKLM\SYSTEM\CurrentControlSet\Services\USBSTOR, Start = 4).

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), NIST SI-3 (Malicious Code Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 2.3 (Address Unauthorized Software)

Compensating: For teams without enterprise endpoint management: deploy the following via Group Policy or local registry on each endpoint. Enable PowerShell Constrained Language Mode by setting HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment__PSLockdownPolicy = 4. Block mshta.exe

using Windows Defender Application Control (WDAC) with a deny rule, or use Software Restriction Policies (SRP) targeting %SystemRoot%\System32\mshta.exe. On the suspected host, immediately run 'netstat -ano | findstr ESTABLISHED' and cross-reference PIDs against LockAppHost.exe via 'tasklist /fi "pid eq "' to confirm anomalous network connections before network isolation. A 2-person team can execute USBSTOR registry change via 'reg add HKLM\SYSTEM\CurrentControlSet\Services\USBSTOR /v Start /t REG_DWORD /d 4 /f' pushed via PSExec or a startup script.

Evidence: Before isolating the host, capture: (1) Full memory image of the LockAppHost.exe process using ProcDump ('procdump -ma LockAppHost.exe lockapphost.dmp') to preserve injected shellcode from DeepLoad's APC injection technique. (2) Current network connection state ('netstat -ano > netstat_snapshot.txt') to document C2 IP/port pairs associated with LockAppHost.exe. (3) WMI subscription state — export 'Get-WMIObject -Namespace root\subscription -Class __EventFilter | Export-Csv wmi_filters_pre.csv' and same for __EventConsumer and __FilterToConsumerBinding — these document the reinfection mechanism before containment disrupts visibility. (4) PowerShell ScriptBlock log from HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon and event log path %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx capturing obfuscated loader commands. (5) Browser credential store file copies: Chrome '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data' and Firefox '%APPDATA%\Mozilla\Firefox\Profiles*.default-release\logins.json' and 'key4.db' to confirm credential theft scope.

Detection — Query WMI event subscriptions for unauthorized entries: run 'Get-WMIObject -Namespace root\subscription -Class __EventFilter', '__EventConsumer', and '__FilterToConsumerBinding' and alert on any entries not matching your baseline. Hunt for PowerShell processes spawned by user-interactive processes (explorer.exe, mshta.exe) with encoded or obfuscated command lines (Event ID 4104 with ScriptBlock logging enabled). Alert on LockAppHost.exe initiating network connections or spawning child processes. Review Windows Security Event ID 4688 for mshta.exe execution with unusual parent processes.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), 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), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For teams without SIEM: deploy Sysmon with a configuration that logs Event ID 1 (Process Create), Event ID 3 (Network Connection), and Event ID 20 (WMI Event Consumer Activity) — SwiftOnSecurity's Sysmon config covers all three. Run the following WMI hunt manually or via scheduled Task on each endpoint: 'Get-WMIObject -Namespace root\subscription -Class __EventFilter | Select Name, Query | Export-Csv C:\IR\wmi_hunt.csv -Append'. Use the free Autoruns (Sysinternals) tool — navigate to the WMI tab — to visually identify non-baseline subscriptions without scripting. For Event ID 4104, enable ScriptBlock logging via GPO at 'Computer Configuration > Administrative Templates > Windows Components > Windows PowerShell > Turn on PowerShell Script Block Logging'. Parse collected .evtx files offline using EvtxECmd (Eric Zimmermann's toolkit) with the command 'EvtxECmd.exe -f Microsoft-Windows-PowerShell%4Operational.evtx --csv C:\IR\ps_scriptblock.csv' and grep for 'FromBase64String' or 'IEX' patterns indicative of DeepLoad's encoded loader stage.

Evidence: Before alerting or modifying state: (1) Export full Windows Security Event Log filtering on Event ID 4688 for mshta.exe parent-child relationships — 'wevtutil qe Security /q:"*[System[EventID=4688]] and *[EventData[Data[@Name='NewProcessName'] and contains(., 'mshta.exe')]]" /f:text > 4688_mshta.txt'. (2) Export Sysmon Event ID 20 (WMI Event Consumer) and Event ID 21 (WMI Event Consumer To Filter) from Microsoft-Windows-Sysmon%4Operational.evtx — these log WMI subscription binding activity specific to DeepLoad's persistence mechanism. (3) Capture the raw WMI repository files before any remediation: '%SystemRoot%\System32\wbem\Repository\OBJECTS.DATA' and 'INDEX.BTR' — these contain subscription objects even if PowerShell queries return empty after partial cleanup. (4) Review ClickFix lure delivery artifacts: browser download history at '%LOCALAPPDATA%\Google\Chrome\User Data\Default\History' (SQLite) for AnyDesk-themed lure filenames, and Windows prefetch at '%SystemRoot%\Prefetch\MSHTA.EXE-*.pf' for execution timestamps.

Eradication — Remove all unauthorized WMI event subscriptions identified in detection step using 'Remove-WMIObject' or WMI Explorer. Terminate and remediate any injected LockAppHost.exe processes. Force password rotation for all credentials stored in Chrome and Firefox on affected hosts — treat stored browser credentials as compromised. Re-image hosts where injection is confirmed rather than attempting in-place cleanup, given the fileless and reinfection characteristics. Verify WMI subscription baseline is clean before declaring eradication; recheck at 72-96 hours post-remediation given the documented reinfection delay.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST IA-5 (Authenticator Management), CIS 5.2 (Use Unique Passwords), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For teams without enterprise reimaging infrastructure: use DISM to capture and restore a known-good OS image ('dism /capture-image /imagefile:C:\Images\baseline.wim /capturedir:C:\ /name:Baseline') if re-imaging is not immediately feasible, but treat any in-place remediation as temporary containment only. To remove identified WMI subscriptions without a commercial tool: 'Get-WMIObject -Namespace root\subscription -Class __FilterToConsumerBinding | Where-Object {$_.Filter -like "**"} | Remove-WMIObject' — execute for all three classes (__EventFilter, __EventConsumer, __FilterToConsumerBinding) in reverse binding order to avoid orphaned objects. Verify WMI repository integrity post-removal by running 'winmgmt /verifyrepository'; if it returns 'WMI repository is inconsistent', run 'winmgmt /resetrepository' only after confirming full memory and subscription exports are preserved. For browser credential invalidation: instruct users to revoke active sessions and rotate passwords at each service (not just locally) since Chrome and Firefox plaintext credential extraction means the credentials are already exfiltrated.

Evidence: Capture before eradication actions destroy forensic state: (1) Full volatile memory image of the affected host using WinPmem ('winpmem_mini_x64.exe output.raw') to preserve APC injection artifacts — injected code within LockAppHost.exe's process memory will not survive reboot or process termination. (2) Named pipe and handle enumeration via Sysinternals Handle.exe ('handle.exe -a > handles_pre_eradication.txt') to document inter-process communication channels established by the injected loader. (3) Chrome 'Login Data' SQLite database copy and Firefox 'logins.json' plus 'key4.db' from all user profiles on the host — these confirm which credentials were stored and therefore potentially exfiltrated. (4) Prefetch files for PowerShell.exe, mshta.exe, and LockAppHost.exe from '%SystemRoot%\Prefetch\' — timestamps establish the attack timeline for the multi-stage loader sequence. (5) WMI repository binary export (OBJECTS.DATA and INDEX.BTR) as documented in detection step — required for determining whether the 72-hour reinfection trigger is time-based or event-based by examining subscription query syntax.

Recovery — Validate WMI subscription baseline matches known-good state at 24, 48, and 96 hours post-remediation. Monitor for re-emergence of obfuscated PowerShell execution or LockAppHost.exe anomalies. Confirm browser credential stores have been cleared and users have rotated passwords for all accounts accessible from affected systems. Verify application control policies blocking mshta.exe and unsigned PowerShell remain enforced after any system updates.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-7 (Least Functionality), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams without continuous monitoring tooling: create a scheduled PowerShell task that runs every 12 hours and exports WMI subscription state to a timestamped CSV, then diffs against the known-good baseline file: 'Compare-Object (Import-Csv C:\IR\wmi_baseline.csv) (Get-WMIObject -Namespace root\subscription -Class __EventFilter | Select Name, Query | Export-Csv C:\IR\wmi_check_$(Get-Date -Format yyyyMMddHHmm).csv -PassThru)'. Schedule this via Task Scheduler as SYSTEM. For obfuscated PowerShell re-emergence: enable Module Logging in addition to ScriptBlock logging (GPO: 'Turn on Module Logging', set module names to '*') and run a daily

'wevtutil qe Microsoft-Windows-PowerShell/Operational /q:"*[EventData[Data contains 'FromBase64']]" /f:text' query. To verify mshta.exe block remains enforced after updates, check WDAC policy effective state with 'CiTool.exe --list-policies' (Windows 11) or verify AppLocker rules via 'Get-AppLockerPolicy -Effective | Test-AppLockerPolicy -Path C:\Windows\System32\mshta.exe'.

Evidence: During recovery monitoring, preserve: (1) Scheduled task output CSVs from WMI subscription checks at 24, 48, and 96-hour marks — any deviation from baseline at the 72-hour window is a confirmed reinfection event requiring immediate return to containment phase. (2) Windows Security Event ID 4688 logs filtered for PowerShell.exe and mshta.exe for the full 96-hour watch period — retain these as evidence of either clean recovery or reinfection activity. (3) Application control event logs from 'Microsoft-Windows-AppLocker%4EXE and DLL.evtx' confirming mshta.exe block enforcement — Event ID 8004 (block) or 8003 (audit) confirm policy is active. (4) Browser credential store state (presence or absence of Login Data and logins.json) verified at recovery completion to confirm user remediation actions were completed.

Post-Incident — This campaign exposed three control gaps: (1) absence of PowerShell ScriptBlock logging and Constrained Language Mode, (2) no WMI subscription monitoring baseline, and (3) reliance on browser-stored credentials as a primary authentication store. Implement NIST SP 800-53 SI-3 (malicious code protection), SI-7 (software integrity), and AU-12 (audit record generation) controls. Evaluate credential manager or password manager deployment to reduce browser credential exposure. Add WMI persistence monitoring to SOC detection backlog as a standing hunt. Brief incident responders on the 72-hour reinfection window — close-out criteria must include WMI validation, not just process termination.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-3 (Malicious Code Protection), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-12 (Audit Record Generation), NIST AU-11 (Audit Record Retention), CIS 8.2 (Collect Audit Logs), CIS 5.2 (Use Unique Passwords), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For teams without a formal lessons-learned process: conduct a 60-minute structured debrief using the three identified gap areas as agenda items. Document action owners for each gap with a 30/60/90-day delivery date. For WMI baseline creation (gap 2): run 'Get-WMIObject -Namespace root\subscription -Class __EventFilter | Export-Csv C:\IR\wmi_baseline_clean.csv' immediately after a confirmed clean rebuild and store this file in a read-only network share — this becomes the diff target for future hunts. For browser credential exposure (gap 3): deploy KeePassXC (free, cross-platform) as an interim password manager with a documented migration guide for users, and push a GPO to disable Chrome's built-in password manager ('PasswordManagerEnabled = false' under HKLM\SOFTWARE\Policies\Google\Chrome) and Firefox equivalent ('signon.rememberSignons = false' via mozilla.cfg). Publish a one-page IR addendum documenting the 72-hour reinfection window so any future responder knows that WMI validation at 96 hours is a hard close-out requirement for any DeepLoad-attributed incident.

Evidence: For the post-incident record, retain: (1) All WMI subscription exports (pre-remediation, post-remediation, and all 96-hour monitoring checkpoints) as the authoritative evidence that the reinfection mechanism was fully removed. (2) The complete PowerShell ScriptBlock event log (Event ID 4104) from Microsoft-Windows-PowerShell%4Operational.evtx spanning the incident timeline — this reconstructs the multi-stage loader sequence (ClickFix lure → mshta.exe → APC injection → LockAppHost.exe) for the incident report. (3) Memory dump artifacts from LockAppHost.exe containing the injected shellcode — submit to a sandbox (any.run, free tier) for behavioral analysis and IOC extraction to feed future detection rules. (4) Browser Login Data SQLite files and Firefox logins.json as evidence of credential theft scope — required for breach notification assessment if regulated data is involved. (5) Documented MITRE ATT&CK mapping: T1566 (Phishing/ClickFix lure), T1218.005 (Signed Binary Proxy Execution: mshta.exe), T1055.004 (Process Injection: APC), T1546.003 (WMI Event Subscription persistence), T1555.003 (Credentials from Web Browsers) — include this mapping in the post-incident report for threat intelligence sharing.

Detection Guidance

Primary detection signals: (1) WMI persistence, query root\subscription namespace for __EventFilter, __EventConsumer, and __FilterToConsumerBinding objects not in your approved baseline; unauthorized entries are high-confidence indicators. (2) PowerShell abuse, enable ScriptBlock Logging (Event ID 4104) and Module Logging; hunt for base64-encoded or heavily obfuscated scripts, especially those spawned from mshta.exe, explorer.exe, or browser processes. (3) Process injection, alert on LockAppHost.exe establishing outbound network connections, spawning child processes, or loading unsigned DLLs; this process has no legitimate network activity profile. (4) LOLBin execution, alert on mshta.exe spawned by user-context processes outside approved administrative workflows (Event ID 4688 with command-line auditing enabled). (5) USB activity, monitor for new removable storage device connections on endpoints via Event ID 2003 (WPD) or Defender for Endpoint device control alerts. (6) Browser credential access, Sysmon Event ID 10 (process access) targeting Chrome or Firefox profile directories (\AppData\Local\Google\Chrome\User Data\Default\Login Data, \AppData\Roaming\Mozilla\Firefox\Profiles\) from unexpected parent processes. Behavioral indicator summary: obfuscated PowerShell + LockAppHost.exe network activity + WMI subscription anomaly constitutes high-confidence DeepLoad activity.

Indicators of Compromise

Type	Value	Context	Confidence
PROCESS	`LockAppHost.exe (with network or child process activity)`	Legitimate Windows process targeted for APC injection; anomalous behavior indicates potential DeepLoad payload execution	MEDIUM
LOLBIN	`mshta.exe`	Used as living-off-the-land binary in DeepLoad execution chain; alert on unexpected parent processes	MEDIUM
FILENAME	`AnyDesk (lure filename)`	DeepLoad uses AnyDesk branding as a social engineering lure; verify any AnyDesk installer against official source and hash	MEDIUM
TECHNIQUE	`WMI event subscription (root\subscription namespace)`	DeepLoad uses WMI subscriptions for autonomous reinfection ~72 hours post-remediation; unauthorized subscriptions are a high-confidence indicator	HIGH
FILEPATH	`\AppData\Local\Google\Chrome\User Data\Default\Login Data`	Chrome credential store targeted by DeepLoad payload; unexpected process access is an indicator of credential theft	MEDIUM
FILEPATH	`\AppData\Roaming\Mozilla\Firefox\Profiles\`	Firefox credential store targeted by DeepLoad payload; unexpected process access is an indicator of credential theft	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1546.003** — Windows Management Instrumentation Event Subscription
- **T1620** — Reflective Code Loading
- **T1566.001** — Spearphishing Attachment
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1140** — Deobfuscate/Decode Files or Information
- **T1218.005** — Mshta
- **T1055.004** — Asynchronous Procedure Call
- **T1059.001** — PowerShell
- **T1539** — Steal Web Session Cookie
- **T1566** — Phishing
- **T1047** — Windows Management Instrumentation
- **T1204.002** — Malicious File
- **T1027** — Obfuscated Files or Information
- **T1176** — Software Extensions
- **T1027.010** — Command Obfuscation
- **T1091** — Replication Through Removable Media
- **T1555.003** — Credentials from Web Browsers

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **CM-3** — Configuration Change Control
- **IA-5** — Authenticator Management
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A03:2021** — Injection

CIS-V8

- **2.5** — Allowlist Authorized Software

- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **16.10** — Apply Secure Design Principles in Application Architectures
- **6.3** — Require MFA for Externally-Exposed Applications
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

ISO-27001-2022

- **A.5.34** — Privacy and protection of personal information

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1546.003	Windows Management Instrumentation Event Subscription	Privilege-Escalation
T1620	Reflective Code Loading	Defense-Evasion
T1566.001	Spearphishing Attachment	Initial-Access
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1218.005	Mshta	Defense-Evasion
T1055.004	Asynchronous Procedure Call	Defense-Evasion
T1059.001	PowerShell	Execution
T1539	Steal Web Session Cookie	Credential-Access
T1566	Phishing	Initial-Access
T1047	Windows Management Instrumentation	Execution
T1204.002	Malicious File	Execution

Technique ID	Technique Name	Tactic
T1027	Obfuscated Files or Information	Defense-Evasion
T1176	Software Extensions	Persistence
T1027.010	Command Obfuscation	Defense-Evasion
T1091	Replication Through Removable Media	Lateral-Movement
T1555.003	Credentials from Web Browsers	Credential-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/03/deepload-malware-uses-clickfix-an...	T3
DeepLoad Malware Pairs ClickFix Delivery with AI-Generated Evasion	https://reliaquest.com/blog/threat-spotlight-deepload-malware-pairs...	T3
Uptick of Malicious PowerShell Processes : r/crowdstrike - Reddit	https://www.reddit.com/r/crowdstrike/comments/1jwpjls/uptick_of_mal...	T3
Browser extensions assessment in Microsoft Defender Vulnerability ...	https://learn.microsoft.com/en-us/defender-vulnerability-management...	T1
http://www.withsecure.com/us/en/support/security-a...	http://www.withsecure.com/us/en/support/security-advisories/cve-202...	T3

Generated 2026-03-30 18:43 UTC by TJS Security Command Center