INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:37 UTC

ClickFix Social Engineering Campaign: Multi-Loader Attack Chain Delivering NetSupport RAT, Latrodectus, and Lumma Stealer

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0101
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Windows systems across multiple industries; abused components include Windows Run dialog, Windows Terminal, MSHTA, jp2launcher.exe (Java Runtime Environment); impersonated brands include DocuSign and Okta
Published	2026-03-24
Discovery Source	Rss

Executive Summary

A social engineering technique called ClickFix has become a primary initial access method in nearly a dozen confirmed incident response engagements in 2025, delivering remote access trojans and credential-stealing malware across multiple industries. Attackers impersonate trusted brands including DocuSign and Okta to trick employees into manually running malicious commands, bypassing email security, endpoint detection, and perimeter controls without any exploit or malicious file attachment. The business risk is direct: successful compromise leads to persistent remote access, credential theft, and potential ransomware staging, with no technical vulnerability to patch.

Technical Analysis

ClickFix is a clipboard-hijacking social engineering technique in which victims are presented with fake CAPTCHA or browser-update prompts on attacker-controlled or compromised sites. The page copies a malicious command to the clipboard and instructs the user to paste and execute it via the Windows Run dialog or Windows Terminal. No CVE applies; the technique exploits user behavior, not a software vulnerability. Three distinct 2025 campaigns have been documented by Palo Alto Unit 42: (1) NetSupport RAT delivered via a new DLL sideloading loader abusing jp2launcher.exe (Java Runtime Environment) through fake DocuSign and GitCode sites, maps to CWE-426 (untrusted search path) and MITRE T1574.002 (DLL side-loading); (2) Latrodectus malware delivered using updated JavaScript obfuscation, distributed via fake browser update lures,

maps to CWE-494 (download of code without integrity check) and T1027 (obfuscated files or information); (3) Lumma Stealer deployed via MSHTA command execution, maps to CWE-77 (command injection via user-pasted commands) and T1218.005 (mshta). Post-execution activity includes keylogging (T1056.001), credential access from browsers and password stores (T1555), screen capture (T1113), and persistence via registry run keys (T1547). Threat actor attribution is unattributed across multiple clusters; SocGholish operators are associated in secondary sources with the NetSupport RAT / fake browser update variant at medium confidence only. No patch exists; the attack surface is entirely behavioral.

Action Checklist

1. Immediate: Block or alert on execution of mshta.exe, jp2launcher.exe, and wscript.exe when launched from user-interactive contexts (Run dialog, Terminal, or browser child processes) via endpoint detection rules or application control policy.
2. Immediate: Push user notification to all staff warning of active fake CAPTCHA and browser-update prompts instructing them to copy and paste commands; include DocuSign and Okta impersonation as named examples.
3. Detection: Search endpoint telemetry for clipboard-paste execution patterns, specifically, command-line arguments to cmd.exe, powershell.exe, or mshta.exe originating from user-interactive parent processes (explorer.exe, Run dialog) within the past 30 days.
4. Detection: Hunt for jp2launcher.exe executing outside of its expected Java installation path, or loading DLLs from non-standard directories (T1574.002); correlate with any outbound connections to unknown hosts from the same process.
5. Assessment: Identify hosts running Java Runtime Environment and confirm jp2launcher.exe is present; assess exposure if DLL sideloading via that binary has not been previously detected or blocked.
6. Assessment: Review web proxy and DNS logs for outbound connections to newly registered domains or domains mimicking DocuSign, Okta, or GitCode; flag any hosts that visited such domains.
7. Communication: Brief SOC analysts, help desk, and IT staff on ClickFix lure characteristics; ensure help desk has a clear escalation path if a user reports being asked to paste a command.
8. Long-term: Implement or audit User Account Control and script execution policies to restrict PowerShell, cmd.exe, mshta.exe, and wscript.exe execution by standard users; evaluate application control (e.g., Windows Defender Application Control or AppLocker) for high-risk binaries.
9. Long-term: Add ClickFix simulation to phishing awareness training programs; standard phishing simulations do not cover clipboard-hijacking lures and will not build the correct user behavior without specific scenario exposure.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive management and consider engaging external IR firm if more than 5 hosts show evidence of command execution via clipboard paste, if jp2launcher.exe DLL sideloading is confirmed in production, or if any exfiltration of credentials or sensitive data to command-and-control servers is detected.

Recovery Notes	After containment and eradication of active threats, conduct a forensic analysis of compromised hosts to determine the extent of post-exploitation activity (lateral movement, credential theft, data exfiltration) using memory dumps, event log timelines, and network flow analysis. Revoke any compromised credentials, reset MFA tokens for affected users, and force password changes via Active Directory. Deploy compensating controls (AppLocker, PowerShell logging, Script Host restrictions) to all endpoints before bringing compromised systems back into production. Conduct a post-incident review with SOC, help desk, and IT leadership to document the attack timeline, detection gaps, response effectiveness, and prioritize long-term controls (application control, UAC hardening, user training).
Forensic Artifacts	Windows Event Log 4688 (Process Creation) and 4690 (Process Termination) with full command-line arguments, parent process details, and timestamps | Sysmon Event Log (Event ID 1 — Process Create, Event ID 3 — Network Connection, Event ID 7 — Image Loaded) for mshta.exe, jp2launcher.exe, cmd.exe, powershell.exe, wscript.exe | Windows Event Log 4657 (Registry Value Set) for HKLM\Software\Microsoft\Windows\CurrentVersion\Run and HKCU registry persistence locations | DNS query logs (Windows Event Log Microsoft-Windows-DNS-Client/Operational, Event ID 3008) and firewall/proxy access logs for outbound connections to newly registered or lookalike domains | Browser history and download artifacts (typically C:\Users\[username]\AppData\Local\Microsoft\Windows\INetCache and C:\Users\[username]\Downloads) for malicious payload storage, including timestamps and source URLs

Per-Action IR Details

Immediate: Block or alert on execution of mshta.exe, jp2launcher.exe, and wscript.exe when launched from user-interactive contexts (Run dialog, Terminal, or browser child processes) via endpoint detection rules or application control policy.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.2.2 (containment strategies)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 AC-6 (Least Privilege), CIS Controls 6.2 (Application Control), CIS Controls 8.5 (Process Monitoring)

Compensating: Use Windows Event Viewer to create automated alerts on Event ID 1 (Process Create) for mshta.exe, jp2launcher.exe, wscript.exe with ParentImage matching explorer.exe, dwm.exe, or rundll32.exe; forward matching events to a centralized text log; use Group Policy to block Script Host execution via gpedit.msc > Computer Configuration > Administrative Templates > Windows Components > Windows Script Host > Prevent execution of scripts (set to Enabled for 64-bit and 32-bit).

Evidence: Before blocking, capture 30 days of Windows Event Log 4688 (Process Creation) and 4690 (Process Termination) records; export parent-child process chains for mshta.exe, wscript.exe, jp2launcher.exe using Get-EventLog or wevtutil; preserve command-line arguments intact (do not truncate); document current baseline of legitimate jp2launcher.exe execution (e.g., Java update paths, parent processes).

Immediate: Push user notification to all staff warning of active fake CAPTCHA and browser-update prompts instructing them to copy and paste commands; include DocuSign and Okta impersonation as named examples.

NIST Phase: Preparation

Reference: NIST 800-61r3 §3.1 (Preparation)

Controls: NIST 800-53 AT-2 (Security Awareness and Training), NIST 800-53 AT-3 (Role-Based Security Training), CIS Controls 15.1 (Security Awareness Program)

Compensating: Distribute a one-page alert via email, intranet, Slack, or Teams with: (1) screenshot or description of ClickFix lure layout (fake CAPTCHA box overlaying login page), (2) exact text phrases observed ('Please verify your

identity', 'Run this command in Terminal'), (3) red flags (asks you to open Run dialog, copy-paste a command, ignore SmartScreen warnings), (4) escalation phone number or ticket system; include examples of impersonated brands (DocuSign, Okta, GitCode); include URL of official incident report if available (e.g., CISA alert, Mandiant reporting).

Evidence: Preserve the alert text and distribution timestamp; document which user populations received notification (scope); capture acknowledgment metrics if platform supports read-receipts; collect any incident reports submitted by users after notification to measure awareness impact.

Detection: Search endpoint telemetry for clipboard-paste execution patterns — specifically, command-line arguments to cmd.exe, powershell.exe, or mshta.exe originating from user-interactive parent processes (explorer.exe, Run dialog) within the past 30 days.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.1 (Detection and Analysis — source identification)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 AU-2 (Audit Events), CIS Controls 8.5 (Process Monitoring)

Compensating: Export Windows Event Log 4688 (Process Creation) from all endpoints for the past 30 days; filter for ProcessName in (cmd.exe, powershell.exe, mshta.exe) with ParentImage in (explorer.exe, rundll32.exe); use command-line argument analysis: search CommandLine for encoded payloads (Base64 strings, -enc switches, IEX, Invoke-WebRequest); manually review each match for telltale lure payloads (certutil.exe -decode, expand, rundll32 with http://, or wmic invocations); correlate timestamps with user logon times to identify interactive sessions; use findstr or grep to extract matches: `wevtutil qe Security /q:*[System[(EventID=4688)]] /rd:true /f:text > processes.txt`; then grep for suspicious patterns.

Evidence: Preserve full command-line arguments (Windows Event Log field: CommandLine) without truncation; capture parent process ID and full process tree (ParentImage, ParentProcessId); record the timestamp and user account (SubjectUserName); document whether SmartScreen warnings appeared (check Windows Defender logs: Microsoft-Windows-Windows Defender/Operational, Event ID 1012-1013); preserve any clipboard history if available (clipboard data is volatile, prioritize collection).

Detection: Hunt for jp2launcher.exe executing outside of its expected Java installation path, or loading DLLs from non-standard directories (T1574.002); correlate with any outbound connections to unknown hosts from the same process.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.1 (Detection and Analysis — attack vector identification)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 CA-7 (Continuous Monitoring), CIS Controls 8.5 (Process Monitoring), CIS Controls 13.1 (Network Traffic Monitoring)

Compensating: Query Windows Event Log 4688 for jp2launcher.exe execution; extract full image path and confirm it matches 'C:\Program Files\Java\jreXXX\bin\jp2launcher.exe' (note: enumerate actual Java versions on assets); use Process Monitor (Sysinternals, free) to capture all DLL load operations by jp2launcher.exe and identify non-standard DLL paths (typically should load from Windows\System32, SysWOW64, and Java installation directory); correlate jp2launcher.exe process ID with Sysmon Event ID 3 (Network Connection) or Windows Event Log 5156 (Outbound network connection allowed) to identify destination IPs/hostnames; cross-reference destinations against known-good Java update servers; use netstat -ano or Get-NetTCPConnection to identify listening and established connections for jp2launcher.exe on compromised host.

Evidence: Capture full process image path and parent process details (Event Log 4688); preserve Sysmon logs (Event ID 7 — Image Loaded) for all DLL loads by jp2launcher.exe including full DLL path, signature status, and signed/unsigned indicator; collect network flow data (Sysmon Event ID 3 or firewall logs) with source port, destination IP, destination port, and protocol; if EDR available, preserve process memory dump and loaded module list; document the Java installation version and expected bin directory path as baseline.

Assessment: Identify hosts running Java Runtime Environment and confirm jp2launcher.exe is present; assess exposure if DLL sideloading via that binary has not been previously detected or blocked.

NIST Phase: Preparation

Reference: NIST 800-61r3 §3.1.3 (Preparation — capability development)

Controls: NIST 800-53 RA-3 (Risk Assessment), NIST 800-53 CM-2 (Baseline Configuration), CIS Controls 2.1 (Hardware Inventory), CIS Controls 2.4 (Software Inventory)

Compensating: Use Group Policy to inventory Java installations: query HKLM\Software\JavaSoft\Java Runtime Environment across all endpoints via remote registry or SCCM; use WMI query (Win32_Product) with filter 'Name LIKE Java%'; for smaller organizations, use Autoruns (Sysinternals, free) exported from each host to search for jp2launcher.exe registrations; scan file system for jp2launcher.exe presence using dir /s /b jp2launcher.exe on each subnet or via network file share enumeration; document total number of hosts with Java installed, version(s), and binary presence; assess whether DLL sideloading mitigations (Side-by-Side manifests, manifest enforcement) are currently deployed for that binary.

Evidence: Document baseline inventory of Java installations (hostname, Java version, installation path, presence/absence of jp2launcher.exe); record which hosts have application control (AppLocker, WDAC) policies applied to Java binaries; preserve registry snapshots of Java runtime configuration (HKLM\Software\JavaSoft) to detect unauthorized modifications; note any previous DLL sideloading detections for this binary in EDR/SIEM (if available) to establish whether protection is already active.

Assessment: Review web proxy and DNS logs for outbound connections to newly registered domains or domains mimicking DocuSign, Okta, or GitCode; flag any hosts that visited such domains.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.1 (Detection and Analysis — attack vector identification)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 AU-2 (Audit Events), CIS Controls 13.1 (Network Traffic Monitoring)

Compensating: Export DNS query logs from Windows Event Log (Microsoft-Windows-DNS-Client/Operational, Event ID 3008 for DNS query success) or DHCP logs; search for domain names containing 'docusign', 'okta', 'gitcode' with suspicious variations (e.g., docusign-verify.com, okta-auth-check.com, gitcode-security.xyz); review web proxy access logs (if available via firewall/proxy appliance) for the same domains; cross-reference DNS queries with outbound connection logs (firewall, Windows Event Log 5156); identify source IP/hostname and timestamp of query; use whois or free online WHOIS tools to check domain registration date (newly registered = higher suspicion); correlate flagged hosts with user account and network segment; create a list of identified lure domains to block at DNS/firewall level.

Evidence: Preserve full DNS query logs with timestamp, query source IP, hostname of source, and resolved destination IP; export web proxy logs including source IP, destination hostname, HTTP status code, request URI, and user-agent string; document any HTTPS connections to flagged domains (which bypass proxy inspection but are logged at DNS level); collect whois registration data and compare against known-good domain registration dates for legitimate brands; flag any connections with geo-location mismatches (e.g., DocuSign domain resolving to IP in unexpected country).

Communication: Brief SOC analysts, help desk, and IT staff on ClickFix lure characteristics; ensure help desk has a clear escalation path if a user reports being asked to paste a command.

NIST Phase: Preparation

Reference: NIST 800-61r3 §3.1.4 (Preparation — tools and resources)

Controls: NIST 800-53 AT-2 (Security Awareness and Training), NIST 800-53 IR-1 (Incident Response Policy)

Compensating: Create a one-page ClickFix reference card for SOC and help desk with: (1) visual example of fake CAPTCHA lure, (2) typical command payload (e.g., mshta.exe, powershell -enc), (3) red flags to watch for, (4) immediate escalation steps (do not let user proceed, capture screenshot if possible, open incident ticket with ticket ID, notify SOC via dedicated Slack channel or email alias); schedule a 30-minute briefing session for SOC analysts and help desk supervisors; provide pre-recorded video or recorded briefing for asynchronous access; include a FAQ document addressing common user questions ('Is Java update safe?', 'Why does the website ask me to run a command?'); distribute contact information for security team and escalation email or phone line.

Evidence: Document training attendance and date; collect feedback/questions from briefing; preserve training materials (slides, reference card, video link); establish a ticket template for ClickFix escalations to ensure consistent data capture; measure help desk effectiveness by tracking escalation volume and time-to-escalation after training

deployment.

Long-term: Implement or audit User Account Control and script execution policies to restrict PowerShell, cmd.exe, mshta.exe, and wscript.exe execution by standard users; evaluate application control (e.g., Windows Defender Application Control or AppLocker) for high-risk binaries.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.2.3 (Recovery — control enhancement) and NIST 800-53 AC-6 (Least Privilege)

Controls: NIST 800-53 AC-6 (Least Privilege), NIST 800-53 AC-3 (Access Control), NIST 800-53 SI-7 (Software, Firmware, and Information Integrity), CIS Controls 6.2 (Application Control), CIS Controls 5.3 (Restrict Administrative Privileges)

Compensating: Use Group Policy to restrict script execution: set HKLM\Software\Policies\Microsoft\Windows\PowerShell\ExecutionPolicy = 'AllSigned' or 'RemoteSigned' and apply gpedit.msc policy 'Computer Configuration > Administrative Templates > Windows Components > Windows PowerShell > Turn on Script Execution'; disable Windows Script Host via gpedit.msc > Computer Configuration > Administrative Templates > Windows Components > Windows Script Host > Prevent execution of scripts (Enabled for both 64-bit and 32-bit); use AppLocker (built-in, free, though requires Windows Enterprise/Education editions) or freeware alternative Autoruns to whitelist only known-good PowerShell and cmd.exe publishers (Microsoft Corporation) and block unsigned/untrusted executables; for organizations without Enterprise Windows, use Windows Defender Exploit Guard in application control mode to block Script Host; pilot policy on test endpoint (10-20 users), measure breakage via help desk tickets over 2 weeks, then phase rollout by department.

Evidence: Document current UAC settings (Registry HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\ConsentPromptBehaviorAdmin); preserve baseline policy audit logs (Event Log 4719 — Policy Change) before and after implementation to track policy application success; capture Windows Defender Application Control events (Event ID 3077-3078 in CodeIntegrity log) or AppLocker events (Event ID 8003-8007) to validate enforcement; record help desk ticket volume for policy-related issues during pilot and rollout phases.

Long-term: Add ClickFix simulation to phishing awareness training programs; standard phishing simulations do not cover clipboard-hijacking lures and will not build the correct user behavior without specific scenario exposure.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §3.2.4 (Post-Incident Activities — lessons learned)

Controls: NIST 800-53 AT-2 (Security Awareness and Training), NIST 800-53 AT-3 (Role-Based Security Training), CIS Controls 15.1 (Security Awareness Program)

Compensating: Develop a ClickFix simulation scenario for your phishing platform (Phish Labs, KnowBe4, Gophish, or internal training module): create an email claiming to be from DocuSign or Okta with a link to a fake login page; on the fake login page, display a fake CAPTCHA or update notification that instructs users to open Terminal/Run and copy-paste a command; do not make the command actually execute — instead, flag the attempt and provide immediate feedback ('You were targeted by a ClickFix attack; do not paste commands from unknown websites'); include educational content on why commands are dangerous; run simulation quarterly with a target of 3-5% user click-through rate (industry baseline is 3-8%); measure improvement over time; track which departments have higher susceptibility and provide targeted follow-up training.

Evidence: Document simulation design and payload (anonymized); record simulation execution dates and duration; capture click-through rates, paste-attempt rates, and reported-to-security rates by department/user group; preserve user feedback and confusion indicators (help desk tickets related to simulation); maintain a control group (subset of users not receiving simulation) to measure knowledge retention impact over time.

Detection Guidance

Primary detection surface is behavioral and post-execution telemetry; perimeter and attachment-based controls will not catch this technique. Key indicators: (1) cmd.exe, powershell.exe, or mshta.exe spawned as children of explorer.exe or with command-line arguments consistent with clipboard-pasted one-liners, look for encoded PowerShell (-EncodedCommand), mshta.exe with a remote URL argument, or curl/certutil downloading to temp directories; (2) jp2launcher.exe executing outside its standard Java installation path or loading unsigned DLLs from %APPDATA%, %TEMP%, or user-writable directories (Windows Security Event 4688 with command-line logging enabled, or EDR process telemetry); (3) outbound HTTP/S connections from mshta.exe, wscript.exe, or jp2launcher.exe to external hosts; (4) NetSupport RAT C2 communication patterns, NetSupport uses a custom protocol over port 443 or 80 and may appear as client32.exe or similarly named processes; (5) Lumma Stealer browser data access, look for processes accessing Chrome, Firefox, or Edge credential stores outside of the browser itself, particularly from recently written executables in temp directories. MITRE techniques to prioritize in detection rules: T1204.002 (malicious file executed by user), T1059.001 (PowerShell), T1059.003 (cmd), T1059.007 (JavaScript), T1218.005 (mshta), T1574.002 (DLL sideloading), T1056.001 (keylogging), T1555 (credential access). No public IOC list is included in available sources for this item; threat intelligence platforms should be queried for current NetSupport RAT, Latrodectus, and Lumma Stealer infrastructure indicators.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`docusign-impersonation lures (specific domains not published in available sources)`	Fake DocuSign sites used to deliver ClickFix clipboard-hijacking prompts and NetSupport RAT; specific domains not available in sources referenced for this item	LOW
DOMAIN	`gitcode-impersonation lures (specific domains not published in available sources)`	Fake GitCode sites used as secondary delivery point for NetSupport RAT via ClickFix; specific domains not available in sources referenced for this item	LOW
HASH	`not available — no file hashes published in sources referenced for this item`	Hashes for NetSupport RAT loader, Latrodectus dropper, and Lumma Stealer samples were not included in the source material available; query threat intelligence platforms (VirusTotal, MalwareBazaar) for current samples	LOW

Framework Mappings

MITRE-ATTACK

- **T1059.003** — Windows Command Shell
- **T1027** — Obfuscated Files or Information
- **T1566** — Phishing
- **T1059.007** — JavaScript
- **T1113** — Screen Capture

- **T1218.011** — Rundll32
- **T1204.002** — Malicious File
- **T1547** — Boot or Logon Autostart Execution
- **T1059.005** — Visual Basic
- **T1574.002** — DLL Side-Loading
- **T1105** — Ingress Tool Transfer
- **T1555** — Credentials from Password Stores
- **T1566.002** — Spearphishing Link
- **T1056.001** — Keylogging
- **T1059.001** — PowerShell
- **T1218.005** — Mshta

NIST-800-53R5

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

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **16.10**
- **2.5**
- **2.6**
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.308(a)(5)(i)** — Security Awareness and Training

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
T1059.003	Windows Command Shell	Execution
T1027	Obfuscated Files or Information	Defense-Evasion
T1566	Phishing	Initial-Access
T1059.007	JavaScript	Execution
T1113	Screen Capture	Collection
T1218.011	Rundll32	Defense-Evasion
T1204.002	Malicious File	Execution
T1547	Boot or Logon Autostart Execution	Persistence
T1059.005	Visual Basic	Execution
T1574.002		
T1105	Ingress Tool Transfer	Command-And-Control
T1555	Credentials from Password Stores	Credential-Access
T1566.002	Spearphishing Link	Initial-Access
T1056.001	Keylogging	Collection
T1059.001	PowerShell	Execution
T1218.005	Mshta	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://unit42.paloaltonetworks.com/preventing-clickfix-attack-vector/	T3
Fake Docusign, Gitcode Sites Spread NetSupport RAT via Multi ...	https://thehackernews.com/2025/06/fake-docusign-gitcode-sites-sprea...	T3

Source	URL	Tier
CAPTCHA to Compromise: Fake DocuSign and Gitcode Sites ...	https://www.linkedin.com/pulse/captcha-compromise-fake-docusign-git...	T3
DocuSign Alert: New Malicious Hacking Tool Mimicking ... - JD Supra	https://www.jdsupra.com/legalnews/docusign-alert-new-malicious-hack...	T3
NetSupport Remote Access Trojan (RAT) delivered through fake ...	https://blogs.opentext.com/netsupport-remote-access-trojan-rat-deli...	T3

Generated 2026-03-29 18:37 UTC by TJS Security Command Center