INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-19 13:50 UTC

BadIIS MaaS Ecosystem: Commodity IIS Malware Toolkit Attributed to Single Developer 'lwxat' Serving Chinese-Speaking Cybercrime Groups

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0338
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Microsoft IIS (Internet Information Services) web servers, all versions susceptible to malicious module injection; Norton antivirus (targeted for AV bypass); Baidu Browser (targeted for traffic compatibility)
Published	2026-05-19T10:00:20+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

A single threat actor known as 'lwxat' has built and commercialized a malware toolkit targeting Microsoft IIS web servers, active since at least September 2021 and observed through January 2026. The toolkit enables silent traffic hijacking, SEO fraud, and reverse proxy abuse, and is sold to multiple Chinese-speaking cybercrime groups, meaning organizations may face several independent actors deploying identical implants. Any organization running internet-facing IIS infrastructure is at risk of persistent compromise that bypasses standard web application controls.

Technical Analysis

BadIIS operates as a malicious IIS native module (.dll injected into the IIS worker process), enabling persistent server-side implantation outside standard web application visibility. The toolkit was attributed to 'lwxat' by Cisco Talos via PDB string analysis and recovery of a builder tool showing active feature branching and reactive AV evasion development from September 2021 through January 2026. Capabilities include traffic redirection (T1102, T1071.001), IIS module persistence (T1505.004), defense evasion targeting Norton AV and Baidu Browser (T1562.001), obfuscation (T1027), masquerading (T1036), and OS/firmware tampering (T1601). The MaaS model (T1588.001, T1583.006, T1583.008) means multiple unrelated operators deploy functionally identical implants, complicating per-incident attribution. No CVE is associated with this campaign; exploitation requires attacker access sufficient to install IIS modules, typically via credential compromise or prior exploitation

of IIS management interfaces. Relevant CWEs: CWE-506 (embedded malicious code), CWE-693 (protection mechanism failure), CWE-326 (inadequate encryption strength). All IIS versions are susceptible to module injection once an attacker achieves the necessary access level. No vendor patch resolves the MaaS toolkit itself; defense relies on hardening, detection, and access control. Primary source: Cisco Talos (https://blog.talosintelligence.com/from-pdb-strings-to-maas-tracking-a-commodity-badiis-ecosystem/).

Action Checklist

1. Detection & Audit: Audit all installed IIS modules immediately on every internet-facing IIS server. Use 'appcmd list module' or the IIS Manager Modules view to enumerate loaded modules. Remove or quarantine any module not explicitly authorized by your application inventory. Disable IIS management interfaces (IIS Manager, Web Deploy) from internet-accessible network paths.
2. Threat Hunting: Query Windows Event Logs (System and Application channels) and IIS logs for unexpected module load events. Hunt for DLLs loaded into w3wp.exe processes that do not match your authorized module baseline. Review HTTP traffic logs for anomalous redirect patterns, unusual referrer chains, or responses that differ between authenticated admin sessions and external user sessions, indicators of selective traffic manipulation.
3. Eradication: There is no vendor patch for BadIIS itself. Remediation requires removing the malicious IIS module DLL, revoking and rotating all credentials with IIS administrative access, and auditing the server for additional persistence mechanisms (scheduled tasks, registry run keys, additional web shells). Rebuild the IIS server from a known-good image if forensic confidence in completeness of removal is low.
4. Recovery: After module removal and credential rotation, validate IIS module inventory against your authorized baseline and confirm no unauthorized modules remain. Monitor outbound HTTP/S traffic from IIS servers for anomalous destinations for a minimum of 30 days post-remediation. Re-enable external access only after a clean module audit is confirmed.
5. Long-term Controls: This campaign exploits the gap between IIS module management controls and standard web application security monitoring. Implement application whitelisting for IIS module loading, enforce least-privilege on IIS service accounts, and integrate IIS module inventory into your change management baseline. Consider deploying integrity monitoring on IIS binary and module directories.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to CISO and legal/compliance if IIS access logs show evidence of traffic hijacking affecting user sessions containing authentication tokens or PII (indicating potential data exposure triggering breach notification obligations), or if multiple IIS servers show simultaneous BadIIS module presence indicating a coordinated MaaS deployment by one or more lwxat customers rather than an isolated compromise.

Recovery Notes	Post-eradication, do not treat IIS server restoration as complete until a signed module inventory audit matches the pre-compromise authorized baseline AND 72 hours of post-recovery outbound traffic monitoring shows no connections to destinations associated with the lwxat reverse proxy or SEO fraud infrastructure. Given that BadIIS is sold as a commodity toolkit to multiple independent Chinese-speaking cybercrime groups, confirm that the initial access vector (IIS management interface exposure, compromised admin credentials, or supply chain) is fully closed — the same access path may be reused by a second buyer of the lwxat toolkit. Maintain elevated IIS module monitoring (automated hash checks every 4 hours) for a minimum of 30 days, as staged persistence mechanisms may attempt to reinstall the module after an apparent successful removal.
Forensic Artifacts	applicationHost.config at %windir%\system32\inetsrv\config\applicationHost.config — the GlobalModules and Modules sections contain the explicit registration entries for BadIIS native modules; any module referencing a DLL path outside the standard inetsrv directory or with a name not matching your authorized baseline is a high-confidence BadIIS indicator specific to the lwxat module injection mechanism. | IIS W3C log files at %SystemDrive%\inetpub\logs\LogFiles\W3SVC*\ — parse for HTTP 301/302 redirect responses correlated with requests bearing Baidu, Sogou, or other Chinese search engine Referer headers; the selective redirect behavior (same URI, different response based on Referer) is the operational signature of the BadIIS traffic hijacking function and distinguishes it from benign redirect configurations. | Sysmon Event ID 7 (Image Loaded) for SourceImage w3wp.exe — captures the exact DLL path and SHA-256 hash of every module loaded into IIS worker processes; a BadIIS module loaded by lwxat's toolkit will appear here with a path or hash not present in your authorized module baseline, and this event type is not generated by default Windows logging without Sysmon. | Memory dump of w3wp.exe processes (via ProcDump -ma) captured before module removal — preserves the in-memory state of the BadIIS implant including any decoded configuration (C2 URLs, redirect rules, target referrer patterns, reverse proxy destinations) that may not be recoverable from the DLL on disk alone, and provides the highest-fidelity artifact for threat intelligence extraction and YARA rule development. | Windows Security Event Log Event ID 4688 (Process Creation) and Event ID 4698/4702 (Scheduled Task Created/Updated) filtered on processes spawned by IIS service accounts or referencing paths in the inetpub or inetsrv directories — the lwxat toolkit has been observed using scheduled tasks for module persistence and reload after removal, making these events the primary artifact for detecting secondary persistence mechanisms installed alongside the core BadIIS module.

Per-Action IR Details

Containment — Audit all installed IIS modules immediately on every internet-facing IIS server. Use 'appcmd list module' or the IIS Manager Modules view to enumerate loaded modules. Remove or quarantine any module not explicitly authorized by your application inventory. Disable IIS management interfaces (IIS Manager, Web Deploy) from internet-accessible network paths.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality) — restrict IIS to only authorized modules, NIST SI-7 (Software, Firmware, and Information Integrity) — verify module integrity before re-enabling, CIS 2.3 (Address Unauthorized Software) — remove unauthorized IIS module DLLs immediately, CIS 4.6 (Securely Manage Enterprise Assets and Software) — enforce module inventory via version-controlled baseline

Compensating: Run: `%windir%\system32\inetsrv\appcmd list module /processModel.userName:*` on each IIS host and diff the output against a saved baseline. If no baseline exists, compare against a clean IIS installation of the same version. Use PowerShell: `Get-WebConfiguration system.webServer/modules/*` to export the full module list to CSV.

Block TCP 8172 (Web Deploy) and TCP 8080/443 for IIS Manager at the host firewall using `netsh advfirewall firewall add rule` to prevent remote module injection while investigation is ongoing. Quarantine (do not delete) suspicious DLLs by moving them to a write-protected staging folder for later analysis.

Evidence: BEFORE removing any module, capture: (1) Full output of `appcmd list module /xml > modules_snapshot.xml` and `Get-WebConfiguration system.webServer/modules/* | Export-Csv` for every IIS site and application pool. (2) File metadata of every DLL in `%windir%\system32\inetsrv\` and `%SystemDrive%\inetpub\` — creation, modification, and last-access timestamps using `Get-ChildItem -Recurse | Select FullName,CreationTime,LastWriteTime,LastAccessTime`. (3) Memory dump of all running w3wp.exe processes via ProcDump (`procdump -ma w3wp.exe`) before any module is removed, to preserve in-memory BadIIS implant state. (4) IIS applicationHost.config (`%windir%\system32\inetsrv\config\applicationHost.config`) — lwxat's modules register as GlobalModules or managed modules here and this file shows the injection point. (5) Hash (SHA-256) of every loaded module DLL for later YARA/VirusTotal comparison.

Detection — Query Windows Event Logs (System and Application channels) and IIS logs for unexpected module load events. Hunt for DLLs loaded into w3wp.exe processes that do not match your authorized module baseline. Review HTTP traffic logs for anomalous redirect patterns, unusual referrer chains, or responses that differ between authenticated admin sessions and external user sessions — indicators of selective traffic manipulation.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring) — monitor w3wp.exe module loads and IIS traffic patterns, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — analyze IIS W3C logs and Windows Event Logs for BadIIS indicators, NIST AU-12 (Audit Record Generation) — ensure IIS extended logging captures sc-bytes, cs-bytes, cs(Referer), and cs(User-Agent) fields, CIS 8.2 (Collect Audit Logs) — enable and retain IIS W3C logs and Windows Application/System event logs, MITRE ATT&CK T1505.004 (IIS Components) — adversary persistence via malicious IIS module registration

Compensating: Deploy Sysmon with a configuration that captures Event ID 7 (Image Loaded) filtered on SourceImage containing 'w3wp.exe' — any DLL load by IIS worker processes not in your authorized list is a BadIIS candidate. Use the following PowerShell to detect selective response manipulation: send identical HTTP requests with and without a 'Referer: https://www.baidu.com/' header to the same URI and diff the responses (`Invoke-WebRequest` with and without `-Headers @{Referer='https://www.baidu.com/'}`) — BadIIS selectively redirects traffic matching Chinese search engine referrers. For log analysis without SIEM, use Log Parser 2.2 (`LogParser.exe -i:W3C 'SELECT cs-uri-stem, sc-status, cs(Referer), COUNT(*) FROM ex*.log WHERE sc-status=302 GROUP BY cs-uri-stem, sc-status, cs(Referer) ORDER BY COUNT(*) DESC'`) to surface anomalous 302 redirect volumes. Write a Sigma rule matching Image Load events on w3wp.exe with ImagePath outside `%windir%\system32\inetsrv\` and `%SystemDrive%\inetpub\`.

Evidence: BEFORE concluding detection scope: (1) IIS W3C log files at `%SystemDrive%\inetpub\logs\LogFiles\W3SVC*\` — parse for HTTP 302/301 redirects with destination domains not matching your application's known domain inventory, and for requests where sc-bytes differs dramatically between similar URI requests (selective content manipulation). (2) Windows System Event Log Event ID 1 (Module Load) and Application Event Log for IIS service events referencing unexpected DLL paths. (3) Sysmon Event ID 7 filtered on w3wp.exe as SourceImage — capture full ImageLoaded path and SHA-256 hash for every DLL. (4) Network flow or proxy logs showing outbound connections from the IIS server's IP to unfamiliar destinations — BadIIS reverse proxy abuse will appear as the IIS server initiating outbound HTTP/S connections it would not normally originate. (5) Compare HTTP response bodies for the same URI served to requests with Baidu/Chinese search engine referrers vs. direct requests — differential responses are the operational fingerprint of this toolkit's traffic hijacking function.

Eradication — There is no vendor patch for BadIIS itself. Remediation requires removing the malicious IIS module DLL, revoking and rotating all credentials with IIS administrative access, and auditing the server for additional persistence mechanisms (scheduled tasks, registry run keys, additional web shells). Rebuild the IIS server from a known-good image if forensic confidence in completeness of removal is low.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling) — execute eradication per documented IR plan, NIST SI-2 (Flaw Remediation) — remove identified malicious components and verify eradication, NIST AC-2 (Account Management) — revoke and rotate all IIS administrative credentials post-compromise, NIST CM-7 (Least Functionality) — restore IIS to minimum authorized module set after eradication, CIS 5.3 (Disable Dormant Accounts) — disable any service or admin accounts used to install the BadIIS module, CIS 7.2 (Establish and Maintain a Remediation Process) — follow risk-based remediation process; prioritize rebuild if eradication confidence is low

Compensating: For credential rotation without enterprise IAM: use `net user` and `Set-LocalUser` to rotate all local IIS admin accounts; if the server is domain-joined, coordinate with AD team to reset any domain accounts with IIS administrative rights and check for newly created accounts (Event ID 4720). For persistence audit without EDR: run `schtasks /query /fo LIST /v > scheduled_tasks.txt` and diff against a known-good export; check `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`, `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`, and `HKLM\SYSTEM\CurrentControlSet\Services` for entries referencing paths in `%SystemDrive%\inetpub\` or temp directories. For web shell detection, use YARA with public web shell rules (e.g., NeoPI, Shell Detector, or ESET's web shell detection YARA rules) scanned against all `.aspx`, `.ashx`, `.asmx`, and `.dll` files in the web root. If confidence in manual eradication is below 100%, treat rebuild as the default, not the fallback.

Evidence: BEFORE eradication, preserve: (1) Full copy of the malicious module DLL(s) with SHA-256 hash and file metadata for threat intelligence sharing and future YARA rule development. (2) Registry export of `HKLM\SOFTWARE\Microsoft\InetStp\Components` and `HKLM\SYSTEM\CurrentControlSet\Services\W3SVC` to document how the module was registered as a service or IIS component. (3) `schtasks /query /fo XML /v > schtasks_full.xml` — lwxat toolkit has been observed installing scheduled tasks for persistence and module reload after removal attempts. (4) Full directory listing with timestamps of `%SystemDrive%\inetpub\`, `%windir%\system32\inetsrv\`, and `%TEMP%` paths on the IIS service account — staging directories used during module deployment. (5) Export of all local and (if applicable) domain accounts with IIS administrative rights and their last logon timestamps to identify the credential used for module installation.

Recovery — After module removal and credential rotation, validate IIS module inventory against your authorized baseline and confirm no unauthorized modules remain. Monitor outbound HTTP/S traffic from IIS servers for anomalous destinations for a minimum of 30 days post-remediation. Re-enable external access only after a clean module audit is confirmed.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling) — verify recovery actions against IR plan before restoring service, NIST SI-7 (Software, Firmware, and Information Integrity) — employ integrity verification on IIS module directories post-recovery, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — maintain elevated monitoring of IIS traffic and module loads for 30 days post-recovery, NIST CP-10 (System Recovery and Reconstitution) — restore system to known-good state and verify operational integrity, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — validate recovered IIS configuration against documented baseline before returning to production

Compensating: For continuous post-recovery module integrity validation without commercial tools: deploy a scheduled PowerShell task running every 4 hours that executes `Get-WebConfiguration system.webServer/modules/*` and compares the hash of each module DLL against a signed baseline CSV — alert on any delta via Windows Event Log write or email. For outbound traffic monitoring without SIEM: configure Windows Firewall logging (`netsh advfirewall set allprofiles logging filename` and `logging droppedconnections enable`) and use a cron-equivalent Scheduled Task to parse firewall logs daily for new outbound destinations from the IIS server's process (filter on w3wp.exe using Sysmon Event ID 3 — Network Connection). The 30-day window is operationally justified by the lwxat toolkit's documented capability for staged persistence — removed modules may be reinstalled by secondary persistence mechanisms not identified during eradication.

Evidence: BEFORE re-enabling external access: (1) Signed, timestamped export of `appcmd list module /xml` from the recovered server as the new clean baseline — store in version control or a write-protected network share. (2) Diff of current `applicationHost.config` against a known-good reference copy — verify GlobalModules and Modules sections

match the authorized baseline exactly. (3) Outbound network connection baseline captured via `netstat -ano` and Sysmon Event ID 3 in the first 24 hours post-recovery, to establish a normal traffic profile before external access is restored. (4) Windows Security Event Log Event ID 4688 (Process Creation) showing w3wp.exe process launch parameters and parent processes post-recovery — confirm no unexpected child processes are spawning from IIS worker processes. (5) Verification that no new DLLs with creation or modification timestamps after the compromise window exist in `%windir%\system32\inetsrv\` or the web root.

Post-Incident — This campaign exploits the gap between IIS module management controls and standard web application security monitoring. Implement application whitelisting for IIS module loading, enforce least-privilege on IIS service accounts, and integrate IIS module inventory into your change management baseline. Consider deploying integrity monitoring on IIS binary and module directories.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity (Lessons Learned)

Controls: NIST IR-4 (Incident Handling) — update IR playbook with BadIIS/lwxat-specific detection and containment procedures, NIST IR-8 (Incident Response Plan) — revise IR plan to include IIS module integrity monitoring as a standing detection requirement, NIST SI-7 (Software, Firmware, and Information Integrity) — implement file integrity monitoring on IIS module directories as a permanent control, NIST CM-7 (Least Functionality) — enforce application whitelisting to prevent unauthorized IIS module registration, NIST AC-6 (Least Privilege) — restrict IIS service account permissions to minimum required; remove local administrator membership, CIS 2.1 (Establish and Maintain a Software Inventory) — add IIS native modules to the authorized software inventory with approved-hash tracking, CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts) — enforce separation between IIS service accounts and administrative accounts used for module management, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — incorporate IIS module audits into recurring vulnerability management cadence, MITRE ATT&CK T1505.004 (IIS Components) — add detection for this technique as a permanent SOC hunt hypothesis

Compensating: Implement AppLocker or Windows Defender Application Control (WDAC) rules restricting DLL loads by IIS Application Pool identities (`IIS AppPool\*`) to an explicit allowlist of approved module paths — this is a free, built-in Windows control that directly blocks the BadIIS module injection mechanism. Deploy osquery with a scheduled query polling `SELECT name, path, sha256 FROM file WHERE path LIKE '%inetsrv%' OR path LIKE '%inetpub%'` every 30 minutes and alerting on new entries. Write a persistent Sigma rule for Sysmon Event ID 7 (ImageLoaded) with SourceImage matching `w3wp.exe` and ImagePath NOT matching an approved module allowlist — this rule should run as a standing hunt query, not a one-time check. For least-privilege IIS service accounts, use `icacls` to explicitly deny WRITE and MODIFY permissions on `%windir%\system32\inetsrv\` to the `IIS_IUSRS` and `IUSR` groups, preventing the service account from being used to install new modules.

Evidence: Post-incident documentation to produce: (1) Lessons-learned report documenting the module name, DLL hash, registration method in applicationHost.config, and behavioral indicators (redirect patterns, referrer-based traffic selection) specific to the lwxat BadIIS variant observed — formatted for sharing with ISACs or CISA. (2) Updated IIS module baseline with SHA-256 hashes of all authorized modules, stored in immutable storage, as the reference document for all future audits. (3) Threat intelligence report linking observed DLL hashes, C2 destinations, and redirect domains to the lwxat MaaS ecosystem for integration into network block lists and future hunting hypotheses. (4) Gap analysis documenting what existing monitoring controls failed to detect the BadIIS module load — specifically whether Sysmon Image Load auditing was absent, whether IIS W3C logs lacked Referer field logging, and whether no change management process existed for applicationHost.config modifications. (5) Updated change management records establishing IIS module changes as a tracked change category requiring approval and hash verification before deployment.

Detection Guidance

Primary detection vectors: (1) IIS module enumeration, run 'appcmd list module /module.name:*' on all IIS hosts and diff against an authorized baseline; unauthorized entries warrant immediate investigation. (2) Process injection hunting, query EDR telemetry for DLLs loaded into w3wp.exe that lack a Microsoft or

application-vendor signature, or whose file paths fall outside standard IIS module directories (typically %windir%\System32\inetsrv). (3) Traffic anomaly analysis, compare HTTP responses served to external clients against responses served to authenticated admin sessions; BadIIS manipulates traffic selectively, so response divergence is a behavioral indicator. (4) Outbound connection review, IIS servers acting as reverse proxies will generate outbound HTTP/S connections inconsistent with their application role; flag w3wp.exe or inetinfo.exe initiating outbound connections to non-internal destinations. (5) File integrity monitoring, alert on new or modified DLLs in IIS module directories without a corresponding authorized change ticket. MITRE techniques to hunt: T1505.004 (IIS components), T1562.001 (impair defenses), T1027 (obfuscated files), T1071.001 (web protocol C2). Cisco Talos has published technical indicators in their original research; review the Talos blog post for any released IOCs and hashes specific to known BadIIS samples.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`Not available in provided source data`	Cisco Talos may have published specific BadIIS sample hashes in their full research post; review https://blog.talosintelligence.com/from-pdb-strings-to-maas-tracking-a-commodity-badiis-ecosystem/ for released indicators	LOW

Framework Mappings

MITRE-ATTACK

- **T1102** — Web Service
- **T1562.001** — Disable or Modify Tools
- **T1071.001** — Web Protocols
- **T1027** — Obfuscated Files or Information
- **T1505.004** — IIS Components
- **T1583.006** — Web Services
- **T1588.001** — Malware
- **T1036** — Masquerading
- **T1601** — Modify System Image
- **T1583.008** — Malvertising

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring

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
T1102	Web Service	Command-And-Control
T1562.001	Disable or Modify Tools	Defense-Evasion
T1071.001	Web Protocols	Command-And-Control
T1027	Obfuscated Files or Information	Defense-Evasion
T1505.004	IIS Components	Persistence
T1583.006	Web Services	Resource-Development
T1588.001	Malware	Resource-Development
T1036	Masquerading	Defense-Evasion
T1601	Modify System Image	Defense-Evasion
T1583.008	Malvertising	Resource-Development

Sources

Source	URL	Tier
Cisco Talos Blog	https://blog.talosintelligence.com/from-pdb-strings-to-maas-trackin...	T3
Microsoft IIS Web Server Discloses Sensitive Information ...	https://www.hkcert.org/security-bulletin/microsoft-iis-web-server-d...	T3
CVE-2020-0645: Microsoft IIS Server Tampering Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2020-0645/	T3
Azure Web App identified target web site is using IIS and ...	https://learn.microsoft.com/en-my/answers/questions/359038/azure-we...	T1
Microsoft IIS Web Server Vulnerabilities | CTF Walkthrough	https://www.youtube.com/watch?v=QtMFd_k9aAg	T3

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 2026-05-19 13:50 UTC by TJS Security Command Center