INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-11 14:58 UTC

DBI versions before 1.650 for Perl read one byte out-of-bounds in preparse when deleting an initial SQL comment

CVE VULNERABILITY | **CRITICAL** | CVSS 9.1

SCC Item ID	SCC-CVE-2026-0411
Type	CVE Vulnerability
CVE ID	CVE-2026-14740
Severity	CRITICAL
CVSS Base Score	9.1
EPSS Score	0.0041 (33th percentile)
Affected Products	DBI versions before 1.650 for Perl; specifically Microsoft azl3 perl-DBI 1.643-5 on Azure Linux 3.0
Published	2026-07-11T01:41:05
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical out-of-bounds read vulnerability (CVE-2026-14740, CVSS 9.1) has been identified in the Perl DBI library versions before 1.650, disclosed as part of Microsoft Patch Tuesday July 2026. Organizations running Azure Linux 3.0 with the azl3 perl-DBI 1.643-5 package are directly affected. If exploited, this flaw could allow an attacker to read sensitive memory contents, potentially exposing credentials or application data processed through Perl database interfaces.

Technical Analysis

CVE-2026-14740 is an out-of-bounds read vulnerability (CWE-125) in the preparse() function of the Perl DBI library. When the parser encounters and attempts to delete an initial SQL comment, it reads one byte beyond the intended buffer boundary. All DBI versions before 1.650 are affected. The specific package confirmed vulnerable is azl3 perl-DBI 1.643-5 on Azure Linux 3.0, disclosed in Microsoft Patch Tuesday July 2026. The vulnerability is mapped to MITRE ATT&CK T1190 (Exploit Public-Facing Application). CVSS base score is 9.1 (Critical); no active exploitation evidence is present in available source data, and the EPSS score of 0.00407 reflects low current exploitation probability. No KEV listing at time of publication. Remediation is upgrade to DBI 1.650 or later. No IOCs or threat actor attribution are associated with this disclosure.

Action Checklist

1. Step 1: Containment, Inventory all systems running Perl DBI versions before 1.650, with immediate focus on Azure Linux 3.0 hosts carrying the azl3 perl-DBI 1.643-5 package. Restrict external network access to any service that exposes Perl DBI parsing to untrusted SQL input until patched. (Note: This is a temporary containment measure; permanent remediation requires patching to DBI 1.650 or later.) Reference: MSRC advisory for CVE-2026-14740 (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-14740).
2. Step 2: Detection, Query package managers across your fleet for installed DBI versions: run 'rpm -qa | grep perl-DBI' on RPM-based systems or 'dpkg -l | grep libdbi-perl' on Debian-based systems. On Azure Linux 3.0, specifically check for azl3 perl-DBI 1.643-5. Review application logs for anomalous SQL parsing errors or memory-related crashes in Perl DBI processes (AU-2: Event Logging; CIS 8.2: Collect Audit Logs). No CVE-specific IOC patterns are available in source data.
3. Step 3: Eradication, Upgrade Perl DBI to version 1.650 or later. On Azure Linux 3.0, apply the updated azl3 perl-DBI package per the Microsoft July 2026 Patch Tuesday release. Validate the upgrade with 'perl -MDBI -e "print $DBI::VERSION"' and confirm output shows 1.650 or higher (CIS 7.3: Perform Automated Operating System Patch Management; CIS 7.4: Perform Automated Application Patch Management).
4. Step 4: Recovery, After patching, restart all Perl application services that depend on DBI. Verify no memory corruption errors appear in application and system logs. Monitor DBI-dependent database query activity for anomalies for at least 72 hours post-remediation (NIST AU-6: Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident, Review your software inventory process to ensure Perl libraries and third-party dependencies are included in vulnerability management scope (CIS 2.1: Establish and Maintain a Software Inventory; CIS 7.1: Establish and Maintain a Vulnerability Management Process). Assess whether automated patch management covers OS-distributed Perl packages on Azure Linux hosts (CIS 7.3). No additional control gaps are identifiable from available source data beyond software inventory and patch cadence.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal/compliance if forensic review of application logs or coredumps reveals evidence of successful out-of-bounds memory reads (e.g., Perl DBI process crashes correlated with externally-sourced SQL comment input), indicating possible credential or sensitive data exfiltration from DBI-connected databases, which may trigger breach notification obligations under applicable data protection regulations.

Recovery Notes	After applying the azl3 perl-DBI upgrade to 1.650+ on all Azure Linux 3.0 hosts and restarting DBI-dependent Perl services, verify each service's loaded DBI version at runtime using 'perl -MDBI -e "print $DBI::VERSION"' and confirm no residual segfault or SIGSEGV entries appear in journalctl for those services. Monitor DBI-backed database query logs and application error logs for at least 72 hours post-patch, baselining against pre-incident query error rates to detect any anomalous SQL parsing failures that could indicate either lingering exploitation attempts or application instability introduced by the version upgrade. If any Perl application service fails to start cleanly after the upgrade, treat it as a potential indicator of a dependency conflict with the new DBI 1.650 API and escalate to the application team before restoring external network access to that service.
Forensic Artifacts	Coredump files under /var/lib/systemd/coredump/ or /var/crash/ for Perl interpreter processes: a successful or attempted exploit of CVE-2026-14740's preparse out-of-bounds read in DBI before 1.650 may produce a SIGSEGV crash, and the coredump captures the exact memory state at the moment of the one-byte over-read, including potentially exposed adjacent memory contents. | Application-layer access logs (e.g., /var/log/httpd/access_log, nginx access.log, or equivalent middleware logs) filtered for requests containing SQL initial comment syntax ('--' or '/*' prefixes in query parameters): these patterns specifically trigger the vulnerable preparse path in DBI versions before 1.650 and would be the attack vector for CVE-2026-14740. | System journal output from 'journalctl -k | grep segfault' scoped to the Perl interpreter binary path: kernel-logged segfaults for the perl process tied to DBI-dependent services provide a timestamped record of potential exploitation events against the azl3 perl-DBI 1.643-5 package on Azure Linux 3.0 hosts. | Output of 'lsof -p | grep \.so' and /proc//maps for running Perl DBI processes at time of discovery: confirms which version of DBI.so (the shared object containing the vulnerable preparse function) was loaded in memory at the time of the incident, and may reveal whether any unexpected or injected shared libraries were loaded alongside it. | RPM transaction history from 'rpm -qa --last | grep perl-DBI' and 'dnf history | grep perl-DBI' on Azure Linux 3.0 hosts: establishes the installation timeline of azl3 perl-DBI 1.643-5 and confirms whether the vulnerable package was present before or after the Microsoft July 2026 Patch Tuesday disclosure window, supporting incident timeline reconstruction.

Per-Action IR Details

Step 1: Containment — Inventory all systems running Perl DBI versions before 1.650, with immediate focus on Azure Linux 3.0 hosts carrying the azl3 perl-DBI 1.643-5 package. Restrict external network access to any service that exposes Perl DBI parsing to untrusted SQL input until patched. Reference: MSRC advisory for CVE-2026-14740 (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-14740).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Run 'rpm -qa | grep perl-DBI' via a bash loop over SSH across all Azure Linux 3.0 hosts (e.g., 'for h in $(cat hosts.txt); do ssh $h rpm -qa | grep perl-DBI; done') to enumerate exposure without a CMDB. Use host-based firewall rules (firewalld or iptables) to block inbound access to ports serving Perl DBI-backed applications (e.g., 'iptables -I INPUT -p tcp --dport -j DROP') on unpatched hosts until the azl3 perl-DBI 1.643-5 package is upgraded.

Evidence: Before restricting network access or modifying firewall rules on any potentially active host, capture: (1) active network connections showing which external IPs are currently reaching Perl DBI-backed services ('ss -tunap' or 'netstat -ano' output); (2) running process list with arguments to identify Perl interpreter processes handling SQL input ('ps auxww | grep perl'); (3) /proc//maps for any active Perl DBI process to record current memory layout and detect anomalous shared library loads. These are destroyed the moment network access is cut or the service is restarted.

Step 2: Detection — Query package managers across your fleet for installed DBI versions: run 'rpm -qa | grep perl-DBI' on RPM-based systems or 'dpkg -l | grep libdbi-perl' on Debian-based systems. On Azure Linux 3.0, specifically check for azl3 perl-DBI 1.643-5. Review application logs for anomalous SQL parsing errors or memory-related crashes in Perl DBI processes (AU-2: Event Logging; CIS 8.2: Collect Audit Logs). No CVE-specific IOC patterns are available in source data.

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), CIS 8.2 (Collect Audit Logs)

Compensating: Use osquery with the query 'SELECT name, version FROM rpm_packages WHERE name LIKE "%perl-DBI%";' across Azure Linux 3.0 hosts to enumerate the vulnerable azl3 perl-DBI 1.643-5 package at scale without a SIEM. For crash detection, parse application logs and system journal for segfault or out-of-bounds signals: 'journalctl -u --since "2026-07-01" | grep -iE "segfault|out.of.bounds|SIGSEGV|killed"'. Write a Sigma rule targeting syslog for Perl process crashes correlated with SQL comment-bearing input strings (initial '--' or '/*' patterns in query logs).

Evidence: Capture before any remediation action: (1) full output of 'journalctl -k | grep -i segfault' to identify any kernel-logged Perl process crashes that may indicate prior exploitation attempts of the CVE-2026-14740 preparse out-of-bounds read; (2) application-layer web server or middleware access logs (e.g., /var/log/httpd/access_log or equivalent) filtered for requests containing SQL comment syntax ('--', '/*') that would trigger the vulnerable preparse path; (3) coredump files under /var/lib/systemd/coredump/ or /var/crash/ for any Perl interpreter crashes, preserving them before patching destroys the pre-patch binary context.

Step 3: Eradication — Upgrade Perl DBI to version 1.650 or later. On Azure Linux 3.0, apply the updated azl3 perl-DBI package per the Microsoft July 2026 Patch Tuesday release. Validate the upgrade with 'perl -MDBI -e "print $DBI::VERSION"' and confirm output shows 1.650 or higher (CIS 7.3: Perform Automated Operating System Patch Management; CIS 7.4: Perform Automated Application Patch Management).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams without automated patch orchestration, script the upgrade across Azure Linux 3.0 hosts using: 'for h in $(cat vulnerable_hosts.txt); do ssh $h "tdnf upgrade perl-DBI -y && perl -MDBI -e \"print \$DBI::VERSION\"" ; done' — capturing per-host version output to a timestamped log file as your upgrade audit trail. Verify package integrity post-install with 'rpm -V perl-DBI' to confirm no file checksum mismatches against the Microsoft-signed azl3 package.

Evidence: Before applying the tdnf upgrade on each Azure Linux 3.0 host, capture: (1) a memory dump or at minimum 'cat /proc//maps' for any running Perl DBI process, since patching will replace the vulnerable DBI.so shared object and destroy the exploitable in-memory state; (2) a snapshot of currently loaded shared libraries for Perl processes ('lsof -p | grep \.so') to confirm which version of DBI.so was loaded at the time of potential exploitation; (3) the pre-patch RPM metadata ('rpm -qi perl-DBI') saved to an evidence file, establishing the pre-remediation baseline for the incident record.

Step 4: Recovery — After patching, restart all Perl application services that depend on DBI. Verify no memory corruption errors appear in application and system logs. Monitor DBI-dependent database query activity for anomalies for at least 72 hours post-remediation (NIST AU-6: Audit Record Review, Analysis, and Reporting).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content Of Audit Records)

Compensating: After restarting Perl DBI-dependent services, set up a continuous log watch using: 'journalctl -fu | grep -iE "segfault|SIGSEGV|out.of.bounds|DBI|SQL"' in a persistent tmux session for the 72-hour monitoring window. Use 'watch -n 300 "ss -tunap | grep "' to spot anomalous new connections to DBI-backed services that might indicate a threat actor probing the previously vulnerable endpoint. Document each service restart with a timestamp and the verified DBI version loaded, confirming the patched azl3 perl-DBI 1.650+ binary is in use.

Evidence: Before restarting Perl application services (which will reload DBI shared libraries and reset process memory state), capture: (1) current application service memory maps ('cat /proc//maps') and open file descriptors ('lsof -p ') for each DBI-dependent Perl process, preserving evidence of the pre-restart loaded library version; (2) any pending or buffered application log output not yet flushed to disk ('journalctl -u --since "" piped to a file) since restart will rotate the in-memory journal buffer. Post-restart, establish a baseline of normal DBI-dependent query rates and error frequencies against which the 72-hour anomaly monitoring is measured.

Step 5: Post-Incident — Review your software inventory process to ensure Perl libraries and third-party dependencies are included in vulnerability management scope (CIS 2.1: Establish and Maintain a Software Inventory; CIS 7.1: Establish and Maintain a Vulnerability Management Process). Assess whether automated patch management covers OS-distributed Perl packages on Azure Linux hosts (CIS 7.3). No additional control gaps are identifiable from available source data beyond software inventory and patch cadence.

NIST Phase: Post Incident

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

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Extend your osquery fleet configuration to continuously report installed Perl module versions: add a scheduled query 'SELECT name, version FROM rpm_packages WHERE name LIKE "%perl%";' with a 24-hour interval so future Perl library CVEs (like CVE-2026-14740 against DBI) surface in your asset inventory automatically without manual scanning. Document the detection gap — that azl3 perl-DBI 1.643-5 was in scope but not covered by existing patch management — as a finding in the lessons-learned report, with a specific action item to add Azure Linux 3.0 tdnf-managed Perl packages to the monthly patch cadence.

Evidence: No volatile evidence capture is required for this post-incident phase. Preserve for the lessons-learned record: (1) the timestamped osquery or RPM scan output that first identified azl3 perl-DBI 1.643-5 as present in the environment, establishing the discovery timeline; (2) the gap analysis artifact showing which Azure Linux 3.0 hosts were absent from the existing software inventory prior to this incident; (3) the pre-patch and post-patch DBI version verification logs produced during Step 3, retained as evidence of remediation completion for any compliance or audit requirement.

Detection Guidance

No CVE-specific IOCs or behavioral indicators are present in the source data. Detection should focus on version identification: run 'rpm -qa | grep perl-DBI' on Azure Linux 3.0 and other RPM-based hosts; any result showing perl-DBI versions below 1.650 indicates exposure. For Debian-based environments, run 'dpkg -l | grep libdbi-perl'. Supplement with application log review for SQL parsing exceptions or abnormal memory errors originating from Perl DBI processes. Per NIST AU-2 and CIS 8.2, ensure audit logging is enabled on hosts running database-connected Perl applications so anomalous query processing is captured. No network-level IOC patterns are available for this vulnerability from current source data.

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-16** — Memory Protection
- **IR-5** — Incident Monitoring

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

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

NIST-CSF-2

- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-14740	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-Jul	T1
CVE-2026-14740 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-14740	T1
CVE Record: CVE-2026-14740	https://www.cve.org/CVERecord?id=CVE-2026-14740	T1
CVE-2026-14740	https://access.redhat.com/security/cve/cve-2026-14740	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 2026-07-11 14:58 UTC by TJS Security Command Center