INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-21 18:38 UTC

# ASP.NET Core Elevation of Privilege Vulnerability (CVE-2026-40372)

CVE VULNERABILITY | CRITICAL | CVSS 9.1

| | |
|---|---|
| SCC Item ID | SCC-CVE-2026-0062 |
| Type | CVE Vulnerability |
| CVE ID | CVE-2026-40372 |
| Severity | CRITICAL |
| CVSS Base Score | 9.1 |
| Affected Products | Microsoft ASP.NET Core 10.0 |
| Published | 2026-04-21T07:00:00 |
| Discovery Source | Msrc Patch Tuesday |

## Executive Summary

Microsoft disclosed a critical elevation of privilege vulnerability in ASP.NET Core 10.0, assigned CVE-2026-40372 with a CVSS score of 9.1, as part of April 2026 Patch Tuesday. An attacker who exploits this flaw could gain elevated system privileges on affected servers, bypassing access controls. Organizations running ASP.NET Core 10.0 in production should treat this as a priority patching event; the out-of-band release of .NET 10.0.7 signals Microsoft assessed the risk as too urgent to hold for a regular update cycle.

## Technical Analysis

CVE-2026-40372 is a critical elevation of privilege (EoP) vulnerability in Microsoft ASP.NET Core 10.0, disclosed April 2026 Patch Tuesday. CVSS base score: 9.1 (Critical). The CVSS vector string should be validated against the MSRC advisory. No CWE classification was confirmed in source data; likely candidates for an EoP class include CWE-285 (Improper Authorization) or CWE-269 (Improper Privilege Management), but these remain unconfirmed and should not be treated as authoritative. MITRE ATT&CK technique T1068 (Exploitation for Privilege Escalation) applies. Microsoft released an out-of-band fix via .NET 10.0.7, indicating the severity warranted immediate action outside the standard patch cycle. EPSS score is 0.0 at time of data capture, reflecting early-stage scoring and should be re-checked as exploitability data matures. The vulnerability is not listed in the CISA KEV catalog as of data capture. No confirmed threat actor attribution or active exploitation evidence was present in the source data. No IOCs are available. Source authority: MSRC (T1), NVD (T1), CISA KEV (T1).

## Action Checklist

1. Step 1: Containment. Identify all systems running ASP.NET Core 10.0 in your environment immediately. Prioritize internet-facing applications. If patching cannot begin within 24 hours, assess whether temporary network-layer restrictions (firewall ACLs, WAF rules) can reduce attack surface for the highest-risk systems while remediation is staged.
2. Step 2: Detection. Query your asset inventory and CMDB for deployments of ASP.NET Core 10.0 (runtime and SDK). Review application server logs for anomalous privilege escalation events, unexpected process spawning under IIS or Kestrel worker processes, and unauthorized access to privileged API endpoints. Correlate with Windows Security Event ID 4672 (Special Privileges Assigned) and 4688 (Process Creation) where applicable. No specific IOC patterns are available for this CVE at this time.
3. Step 3: Eradication. Apply the .NET 10.0.7 out-of-band security update to all affected systems. Reference the MSRC Update Guide entry for CVE-2026-40372 (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-40372) for authoritative patch guidance. Follow your organization's change management process; given CVSS 9.1, an emergency change window is appropriate.
4. Step 4: Recovery. After applying .NET 10.0.7, verify the installed runtime version on all patched systems using 'dotnet --list-runtimes' and confirm the update appears in Windows Update history or deployment tooling logs. Monitor privileged account activity and application error logs for 72 hours post-patch. Validate that application functionality is intact following the update.
5. Step 5: Post-Incident. Review whether your vulnerability management program has a defined SLA for critical out-of-band patches (CVSS 9.0+). If not, establish one. Assess whether your asset inventory would have identified all ASP.NET Core 10.0 deployments quickly; gaps here increase exposure windows. Review least-privilege configurations on ASP.NET Core application service accounts to limit the blast radius of any future EoP vulnerability.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | IMMEDIATE |
| **Escalation Criteria** | Escalate to CISO and initiate formal incident declaration if Windows Security Event ID 4672 shows SeDebugPrivilege or SeTcbPrivilege assigned to any ASP.NET Core application pool service account identity, or if IIS/Kestrel process logs show cmd.exe, powershell.exe, or net.exe spawned as child processes of w3wp.exe or dotnet.exe — either condition indicates likely active exploitation of CVE-2026-40372 and triggers breach notification assessment if the affected application processes PII or PHI. |

| | |
|---|---|
| **Recovery Notes** | After applying .NET 10.0.7, verify the runtime replacement on every patched host using 'dotnet --list-runtimes' and validate that no residual 10.0.x pre-patch runtime remains active in any IIS application pool by recycling all pools and confirming they load against 10.0.7 in the application event log. Maintain continuous monitoring of Windows Security Event IDs 4672 and 4688 for all IIS worker process accounts for a minimum of 72 hours post-patch, as any attacker who achieved privilege escalation during the exposure window may have established persistence via scheduled tasks, registry run keys, or service installations that survive the runtime update. If IIS application functionality regression is observed post-patch, test against the .NET 10.0.7 release notes compatibility matrix before rolling back, as rollback would re-expose the CVE-2026-40372 attack surface. |
| **Forensic Artifacts** | Windows Security Event Log — Event ID 4672 (Special Privileges Assigned) for IIS application pool service account SIDs (e.g., IIS APPPOOL\) gaining SeDebugPrivilege, SeTcbPrivilege, or SeImpersonatePrivilege, which directly reflects the privilege escalation outcome of CVE-2026-40372 exploitation \| Sysmon Event ID 1 (Process Creation) logs showing process ancestry chains where cmd.exe, powershell.exe, whoami.exe, or net.exe are spawned as children of w3wp.exe (IIS) or dotnet.exe (Kestrel), indicating code execution under an escalated context achieved via the ASP.NET Core 10.0 EoP flaw \| IIS access logs at C:\inetpub\logs\LogFiles\W3SVC* — specifically requests to ASP.NET Core middleware pipeline endpoints (authorization, authentication, identity routes) that transitioned from HTTP 401/403 to HTTP 200 without a corresponding legitimate authentication event, suggesting authorization bypass consistent with this EoP class \| ASP.NET Core application stdout/stderr logs and Windows Application Event Log entries from .NET runtime source for System.Security.SecurityException or Microsoft.AspNetCore.Authorization exceptions immediately followed by successful privileged operations — the exception-then-success pattern may indicate exploitation probing or a successful bypass of the authorization middleware \| Pre- and post-patch dotnet.exe and Microsoft.AspNetCore.App runtime DLL file hashes from C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App\10.0.x\ — deviation from Microsoft-published hashes for 10.0.7 after patching would indicate tampering with the runtime binaries as a persistence mechanism following exploitation |

### Per-Action IR Details

**Step 1: Containment — Identify all systems running ASP.NET Core 10.0 in your environment immediately. Prioritize internet-facing applications. If patching cannot begin within 24 hours, assess whether temporary network-layer restrictions (firewall ACLs, WAF rules) can reduce attack surface for the highest-risk systems while remediation is staged.**

**NIST Phase:** Containment

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

**Controls:** NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers)

**Compensating:** Run 'dotnet --list-runtimes' remotely across Windows hosts via PowerShell: Invoke-Command -ComputerName (Get-Content servers.txt) -ScriptBlock { dotnet --list-runtimes } | Select-String '10.0'. On Linux, use: pdsh -w ^hosts.txt 'dotnet --list-runtimes | grep 10.0'. For internet-facing IIS hosts, immediately add a WAF rule blocking requests with anomalous Authorization headers or unusual HTTP verb combinations while patch staging occurs. Use Windows Firewall (netsh advfirewall) to restrict inbound 443/80 to known IP ranges on non-public-facing ASP.NET Core 10.0 hosts as a temporary measure.

**Evidence:** Before restricting network access, capture a netstat snapshot ('netstat -ano' on Windows, 'ss -tulnp' on Linux) to document all active connections to IIS or Kestrel process PIDs hosting ASP.NET Core 10.0 applications. Export current IIS application pool identity configurations from IIS Manager or via 'appcmd list apppool /processModel.userName:*' to establish a privilege baseline. Preserve Windows Security Event Log state — export the

current Security log (wevtutil epl Security pre-containment-baseline.evtx) before any ACL changes alter subsequent logging.

**Step 2: Detection — Query your asset inventory and CMDB for deployments of ASP.NET Core 10.0 (runtime and SDK). Review application server logs for anomalous privilege escalation events, unexpected process spawning under IIS or Kestrel worker processes, and unauthorized access to privileged API endpoints. Correlate with Windows Security Event ID 4672 (Special Privileges Assigned) and 4688 (Process Creation) where applicable. No specific IOC patterns are available for this CVE at this time.**

**NIST Phase:** Detection Analysis

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

**Controls:** NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content of Audit Records), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

**Compensating:** Deploy Sysmon with SwiftOnSecurity config (sysmonconfig-export.xml) and focus on Event ID 1 (Process Create) to detect child processes spawned by w3wp.exe (IIS) or the dotnet runtime process that are anomalous — e.g., cmd.exe, powershell.exe, net.exe, or whoami.exe. Query with: Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 1 -and $_.Message -match 'w3wp.exe'}. For Event ID 4672, run: Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4672} | Where-Object {$_.Message -notmatch 'SYSTEM|LOCAL SERVICE|NETWORK SERVICE'} to surface non-standard privilege assignments. Cross-reference with IIS logs at C:\inetpub\logs\LogFiles\W3SVC* for HTTP 500 errors or unusual URI patterns targeting authentication or authorization endpoints in the affected ASP.NET Core 10.0 application.

**Evidence:** Collect and preserve: (1) Windows Security Event Log filtered for Event IDs 4672 and 4688 in the 30-day window preceding detection, exported via 'wevtutil epl Security detection-window.evtx'. (2) IIS access logs from C:\inetpub\logs\LogFiles\W3SVC* for all vhosts running ASP.NET Core 10.0 — look for requests to middleware pipeline endpoints or identity/authorization routes returning unexpected 200s after prior 401/403s. (3) Sysmon Event ID 1 logs showing process ancestry from w3wp.exe or dotnet.exe host processes. (4) Application event logs from Windows Event Viewer under 'Application' source for .NET runtime exceptions, particularly System.Security or System.UnauthorizedAccessException entries that may indicate failed or successful privilege boundary crossing. (5) If Kestrel is used as the edge server, capture stdout/stderr application logs from the ASP.NET Core application's configured log output path for entries referencing middleware authorization failures or role claim tampering.

**Step 3: Eradication — Apply the .NET 10.0.7 out-of-band security update to all affected systems. Reference the Microsoft Developer Blog advisory (devblogs.microsoft.com/dotnet/dotnet-10-0-7-oob-security-update/) and the MSRC Update Guide entry for CVE-2026-40372 for authoritative patch guidance. Follow your organization's change management process; given CVSS 9.1, an emergency change window is appropriate.**

**NIST Phase:** Eradication

**Reference:** NIST 800-61r3 §3.4 — Eradication

**Controls:** NIST SI-2 (Flaw Remediation), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST CM-7 (Least Functionality), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process)

**Compensating:** For teams without enterprise patch management, use winget to deploy the update on Windows: 'winget upgrade Microsoft.DotNet.Runtime.10 --version 10.0.7'. For Linux hosts, use the Microsoft package feed: 'sudo apt-get install --only-upgrade dotnet-runtime-10.0=10.0.7' (Debian/Ubuntu) or 'sudo dnf upgrade dotnet-runtime-10.0-10.0.7' (RHEL/Fedora). Automate across a fleet with a simple Ansible playbook targeting the dotnet package name. Verify the update does not break application startup by running the application under a staging IIS app pool or with 'dotnet run' and reviewing output for startup exceptions before promoting to production pools.

**Evidence:** Before applying .NET 10.0.7, snapshot the current runtime version on each target system: 'dotnet --list-runtimes > pre-patch-runtimes.txt' and hash the dotnet.exe binary ('Get-FileHash C:\Program Files\dotnet\dotnet.exe'). Preserve the current GAC state and module list from the affected IIS worker process via Task Manager or 'Get-Process w3wp | Select-Object Modules' to establish a clean baseline for post-patch comparison. If exploitation is suspected (not just exposure), take a full memory image of the w3wp.exe process using ProcDump

('procdump -ma w3wp.exe w3wp_pre-patch.dmp') before patching, as eradication will overwrite runtime binaries that may contain evidence of in-memory manipulation.

**Step 4: Recovery — After applying .NET 10.0.7, verify the installed runtime version on all patched systems using 'dotnet --list-runtimes' and confirm the update appears in Windows Update history or deployment tooling logs. Monitor privileged account activity and application error logs for 72 hours post-patch. Validate that application functionality is intact following the update.**

**NIST Phase:** Recovery

**Reference:** NIST 800-61r3 §3.5 — Recovery

**Controls:** NIST SI-7 (Software, Firmware, and Information Integrity), NIST SI-2 (Flaw Remediation), NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 2.2 (Ensure Authorized Software is Currently Supported)

**Compensating:** Post-patch, run integrity verification: hash the updated dotnet.exe and core runtime DLLs ('Get-FileHash "C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App\10.0.7*" -Algorithm SHA256 > post-patch-hashes.txt') and compare against Microsoft's published file hashes in the .NET 10.0.7 release notes. Set a 72-hour Sysmon watch on Event ID 1 for child process creation from w3wp.exe and Event ID 3 (Network Connection) from the dotnet runtime process to detect any persistence mechanism that survived patching. Use osquery to schedule a recurring check: 'SELECT * FROM processes WHERE name = "w3wp.exe" OR name = "dotnet.exe"' and alert on new child processes during the monitoring window.

**Evidence:** Capture post-patch state: 'dotnet --list-runtimes > post-patch-runtimes.txt' and diff against the pre-patch snapshot to confirm 10.0.7 is installed and 10.0.x (pre-patch) is no longer the active runtime. Export Windows Update history ('Get-HotFix | Where-Object {$_.InstalledOn -gt (Get-Date).AddDays(-1)}') to document the patch application timestamp for change management records. For the 72-hour monitoring window, continuously collect Windows Security Event ID 4672 events and filter for application pool service account SIDs gaining SeDebugPrivilege or SeTcbPrivilege — either would indicate the EoP condition persists or a persistence mechanism is active post-patch.

**Step 5: Post-Incident — Review whether your vulnerability management program has a defined SLA for critical out-of-band patches (CVSS 9.0+). If not, establish one. Assess whether your asset inventory would have identified all ASP.NET Core 10.0 deployments quickly — gaps here increased exposure windows. Review least-privilege configurations on ASP.NET Core application service accounts to limit the blast radius of any future EoP vulnerability.**

**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 RA-3 (Risk Assessment), NIST SI-2 (Flaw Remediation), NIST CM-7 (Least Functionality), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

**Compensating:** Implement a recurring osquery scheduled query to detect new ASP.NET Core runtime installations: 'SELECT * FROM programs WHERE name LIKE "%ASP.NET Core%" OR name LIKE "%.NET Runtime%"' exported weekly and diffed against a known-good baseline — this closes the asset visibility gap exposed by this incident. For service account least-privilege review, audit IIS application pool identities via 'appcmd list apppool /processModel.userName:*' and ensure each pool runs under a dedicated low-privilege gMSA (Group Managed Service Account) rather than a shared or elevated account. Document a formal emergency patch SLA: CVSS 9.0+ out-of-band releases must be assessed within 4 hours and patched within 24 hours on internet-facing systems.

**Evidence:** For the lessons-learned record, preserve the full timeline artifact set: pre-patch runtime inventory, patch deployment logs with timestamps, Event ID 4672/4688 exports from the exposure window, and the IIS access log archive covering the period between CVE-2026-40372 disclosure and patch completion. This constitutes the evidentiary record demonstrating due care and should be retained per NIST AU-11 (Audit Record Retention) for the organization's defined retention period. If any exploitation activity was confirmed or suspected during the exposure window, escalate artifact preservation to forensic-quality chain-of-custody handling before archiving.

## Detection Guidance

No confirmed exploitation indicators or IOCs are available for CVE-2026-40372 at this time. Detection should focus on anomaly-based signals. On Windows hosts running ASP.NET Core 10.0 under IIS: monitor Windows Security Event ID 4672 (Special Privileges Assigned to New Logon) and 4688 (A New Process Has Been Created) for unexpected privilege grants tied to IIS worker processes (w3wp.exe) or dotnet.exe. On Linux hosts using Kestrel: review application logs for unexpected process privilege changes or unauthorized syscall patterns (auditd rules targeting setuid/setgid calls). Query your SIEM for ASP.NET Core 10.0 hosts that generate privilege-related events after the vulnerability disclosure date (April 2026) but before patch application. Re-check EPSS score (currently 0.0, expected to update as exploitability data matures) and monitor CISA KEV catalog for addition of this CVE, which would indicate confirmed active exploitation.

## Framework Mappings

### MITRE-ATTACK

- **T1068** — Exploitation for Privilege Escalation

### NIST-800-53R5

- **AC-6** — Least Privilege
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation

### CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **5.4** — Restrict Administrator Privileges to Dedicated Administrator Accounts

### ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1068 | Exploitation for Privilege Escalation | Privilege-Escalation |

## Sources

| Source | URL | Tier |
|---|---|---|
| **MSRC Update Guide** | https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-40372 | T1 |
| **(consolidated)** | https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-Apr | T1 |

| Source | URL | Tier |
|---|---|---|
| **.NET 10.0.7 Out-of-Band Security Update - Microsoft Developer Blogs** | https://devblogs.microsoft.com/dotnet/dotnet-10-0-7-oob-security-up... | T1 |
| **Known Exploited Vulnerabilities Catalog \| CISA** | https://www.cisa.gov/known-exploited-vulnerabilities-catalog | T1 |
| **NVD - Search and Statistics** | https://nvd.nist.gov/vuln/search | T1 |
| **NVD** | https://nvd.nist.gov/vuln/detail/CVE-2026-40372 | T1 |



Generated 2026-04-21 18:38 UTC by TJS Security Command Center