INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-08 14:01 UTC

# Critical RCE Vulnerabilities in Microsoft Semantic Kernel AI Agent Framework

CVE VULNERABILITY | CRITICAL

| | |
|---|---|
| SCC Item ID | SCC-CVE-2026-0147 |
| Type | CVE Vulnerability |
| CVE ID | CVE-2026-25592, CVE-2026-26030 |
| Severity | CRITICAL |
| EPSS Score | 0.0007 (21th percentile) |
| Affected Products | Microsoft Semantic Kernel framework (specific versions not confirmed from available data) |
| Published | 2026-05-07 |
| Discovery Source | Gemini |

## Executive Summary

Microsoft disclosed two critical Remote Code Execution vulnerabilities (CVE-2026-25592, CVE-2026-26030) in its Semantic Kernel AI agent framework, affecting organizations that have built AI-powered applications on this platform. An attacker who can inject malicious prompts into an AI agent may be able to execute arbitrary code on the underlying system. Organizations deploying Semantic Kernel in production AI workflows face direct risk of system compromise and potential lateral movement within their environment. Affected versions are not yet confirmed from available data; verify against the Microsoft Security Response Center advisory before scoping your response.

## Technical Analysis

CVE-2026-25592 is characterized as a path traversal flaw (CWE-22) in the Microsoft Semantic Kernel framework. CVE-2026-26030 technical mechanism is not confirmed from available data; treat as Remote Code Execution pending direct review of the MSRC advisory. Both vulnerabilities are reported to be exploitable via prompt injection attacks against AI agents built on the framework, mapping to CWE-77 (command injection). Successful exploitation may allow unauthorized Remote Code Execution on the host system. MITRE ATT&CK coverage: T1203 (Exploitation for Client Execution), T1190 (Exploit Public-Facing Application), T1059 (Command and Scripting Interpreter). CVSS base scores are not yet published in available data (reported as 0.0, treat as pending NVD publication). EPSS score: 0.00067 (20th percentile), indicating low current exploitation probability but not zero risk given the RCE classification. CISA KEV: not listed as of available data. Affected versions are not confirmed from available data; direct review of the Microsoft Security Response

Center advisory and NVD records is required. Primary source URLs are present in the sources section; content was not directly accessed during this analysis. All technical specifics should be verified against primary sources before operational decisions are made.

## Action Checklist

1. Step 1: Containment, Identify all internal applications and pipelines built on Microsoft Semantic Kernel. Isolate or restrict network access to services running the framework until affected versions are confirmed and patched. Treat any internet-facing AI agent endpoint as high priority. Consult the Microsoft Security Response Center advisory for CVE-2026-25592 and CVE-2026-26030 to determine affected version ranges before scoping containment.
2. Step 2: Detection, Review logs on hosts running Semantic Kernel for anomalous file path traversal patterns (sequences containing '../' or encoded equivalents) in prompt input fields or API request bodies. Monitor for unexpected child process spawning from the Semantic Kernel process, unusual outbound connections, and any file reads or writes outside expected application directories. Check application logs for prompt content that includes shell metacharacters or script fragments. Specific event IDs and log query syntax will depend on your logging stack; correlate against T1059 and T1203 behavioral patterns in your SIEM.
3. Step 3: Eradication, Apply the patch or version upgrade specified in the Microsoft Security Response Center advisory once affected version ranges are confirmed. Do not assume a specific version is safe until confirmed by the vendor advisory. If patching is not immediately possible, restrict Semantic Kernel agent endpoints to trusted internal callers only and implement input validation on all prompt ingestion points to block path traversal sequences and command injection patterns.
4. Step 4: Recovery, After patching, verify the installed Semantic Kernel version matches the remediated version listed in the Microsoft advisory. Re-enable any isolated services incrementally, monitoring for the same anomalous behaviors identified in the detection step. Confirm no unauthorized files were written or modified during any potential exploitation window by auditing file system changes on affected hosts.
5. Step 5: Post-Incident, Review your AI agent architecture for prompt injection attack surface broadly, not only in Semantic Kernel. Assess whether user-controlled input ever reaches framework-level execution without sanitization. Map this gap to NIST SP 800-53 SI-10 (Information Input Validation) and consider adding AI-specific threat modeling to your secure development lifecycle for any future AI agent projects.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | IMMEDIATE |
| **Escalation Criteria** | Escalate to senior IR leadership, legal, and privacy counsel immediately if forensic analysis confirms that a Semantic Kernel agent process spawned an unexpected child process, wrote files outside its application directory, or established outbound connections during the exposure window — any of which indicates successful RCE that may have exposed data processed by the AI agent, triggering breach notification assessment under GDPR, HIPAA, or applicable state privacy law. |

| | |
|---|---|
| **Recovery Notes** | After patching, maintain enhanced Sysmon process creation and network connection logging on all formerly isolated Semantic Kernel hosts for a minimum of 30 days, watching specifically for dotnet.exe spawning interpreter processes or initiating outbound connections to non-whitelisted destinations, which would indicate a persistent implant survived remediation. Re-enable internet-facing AI agent endpoints last and only after internal endpoints have been stable for 24-48 hours under active monitoring. Verify that all Semantic Kernel NuGet package references across CI/CD pipelines and container images have been updated to the patched version, not only the production runtime, to prevent reintroduction of the vulnerable library through a deployment. |
| **Forensic Artifacts** | IIS/Kestrel/NGINX access logs for Semantic Kernel API endpoints: POST request bodies containing '../', '%2e%2e%2f', '%252e', shell metacharacters (';', '\|', '`', '$('), or base64-encoded strings — the primary delivery artifact for prompt injection RCE exploiting CVE-2026-25592 and CVE-2026-26030. \| Sysmon Event ID 1 (Process Create) logs filtered on parent process 'dotnet.exe': any child process of 'cmd.exe', 'powershell.exe', 'sh', 'bash', 'python', or 'curl' spawned from the Semantic Kernel host process is a high-confidence RCE indicator for this vulnerability class (MITRE T1203, T1059). \| Filesystem modification timeline on affected hosts scoped to the Semantic Kernel application root, system temp directories ('C:\Windows\Temp', '/tmp', '/var/tmp'), and web-accessible directories: unexpected script files (.ps1, .sh, .py, .aspx, .php) written during the exposure window indicate successful post-exploitation file staging. \| Application 'packages.lock.json' and '.deps.json' files from all .NET project deployment directories: these record the exact resolved Semantic Kernel NuGet version at runtime and are the definitive evidence for confirming whether a host was running a vulnerable version of the framework at time of exploitation. \| Network flow logs or Sysmon Event ID 3 (Network Connect) records for the Semantic Kernel service process PID: unexpected outbound connections from dotnet.exe to non-whitelisted external IPs or unusual high-numbered ports during the exposure window indicate C2 callback activity following successful RCE exploitation. |

## Per-Action IR Details

**Step 1: Containment — Identify all internal applications and pipelines built on Microsoft Semantic Kernel. Isolate or restrict network access to services running the framework until affected versions are confirmed and patched. Treat any internet-facing AI agent endpoint as high priority. Check the Microsoft Security Response Center advisory (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-25592) for affected version ranges before scoping containment.**

**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:** Use 'netstat -ano' or 'ss -tulnp' to enumerate listening services and correlate PIDs to Semantic Kernel .NET processes (dotnet.exe on Windows, dotnet on Linux). Cross-reference with 'tasklist /svc' or 'ps aux | grep dotnet'. Block inbound traffic to identified Semantic Kernel API ports at the host firewall using 'netsh advfirewall firewall add rule' (Windows) or 'iptables -I INPUT -p tcp --dport -j DROP' (Linux) until patch status is confirmed. For CI/CD pipelines consuming Semantic Kernel NuGet packages, inspect project .csproj or packages.lock.json files with 'grep -r "SemanticKernel" /path/to/repo' to enumerate exposure.

**Evidence:** Before isolating any host, capture: (1) a full list of running processes and their parent-child relationships via 'Get-Process | Select-Object Name,Id,Path | Export-Csv' or 'ps auxf' to baseline what was running at containment time; (2) active network connections from the Semantic Kernel service process using 'netstat -anob' (Windows) or 'ss -tulnp' (Linux) to document any established outbound connections that may indicate exploitation already occurred; (3) the installed Semantic Kernel NuGet package version from the application's 'packages.lock.json', '.deps.json', or

'obj/project.assets.json' to confirm whether the host is within the vulnerable version range before scoping containment effort.

**Step 2: Detection — Review logs on hosts running Semantic Kernel for anomalous file path traversal patterns (sequences containing '../' or encoded equivalents) in prompt input fields or API request bodies. Monitor for unexpected child process spawning from the Semantic Kernel process, unusual outbound connections, and any file reads or writes outside expected application directories. Check application logs for prompt content that includes shell metacharacters or script fragments. Specific event IDs and log query syntax will depend on your logging stack; correlate against T1059 and T1203 behavioral patterns in your SIEM.**

**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-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

**Compensating:** Deploy Sysmon with a configuration that captures Event ID 1 (Process Create) to detect child processes spawned by 'dotnet.exe' (the Semantic Kernel host process) — specifically watch for 'cmd.exe', 'powershell.exe', 'sh', 'bash', or 'python' as children of the .NET runtime process. Use Sysmon Event ID 11 (File Create) and Event ID 3 (Network Connect) filtered on the dotnet PID to detect out-of-directory file writes and unexpected outbound connections. For web-facing Semantic Kernel APIs, parse IIS logs or Kestrel application logs using 'Select-String -Path C:\inetpub\logs\LogFiles* -Pattern "\.\./|%2e%2e%2f|%252e|cmd=|;whoami|powershell"' to surface path traversal and command injection patterns consistent with CVE-2026-25592 and CVE-2026-26030 exploitation. Map findings to MITRE ATT&CK T1059 (Command and Scripting Interpreter) and T1203 (Exploitation for Client Execution).

**Evidence:** Preserve before analysis: (1) IIS/Kestrel/NGINX access logs covering the 30-day window prior to detection, specifically POST request bodies to Semantic Kernel agent API endpoints (e.g., '/api/chat', '/api/agent', '/invoke') that contain '../', '%2e%2e', '|', ';', '`', '$(' or base64-encoded payloads — these are the primary delivery vector for prompt injection RCE in this CVE class; (2) Sysmon Event ID 1 logs showing process creation chains rooted at 'dotnet.exe' — any shell or interpreter spawned from the .NET host process is a high-confidence indicator of T1203 exploitation; (3) Windows Security Event Log Event ID 4688 (Process Creation with command line auditing enabled) filtered on parent processes matching the Semantic Kernel service to catch post-exploitation command execution consistent with T1059.

**Step 3: Eradication — Apply the patch or version upgrade specified in the Microsoft Security Response Center advisory once affected version ranges are confirmed. Do not assume a specific version is safe until confirmed by the vendor advisory. If patching is not immediately possible, restrict Semantic Kernel agent endpoints to trusted internal callers only and implement input validation on all prompt ingestion points to block path traversal sequences and command injection patterns.**

**NIST Phase:** Eradication

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

**Controls:** NIST SI-2 (Flaw Remediation), NIST SI-10 (Information Input Validation), NIST CM-7 (Least Functionality), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management)

**Compensating:** Update the Semantic Kernel NuGet package to the remediated version using 'dotnet add package Microsoft.SemanticKernel --version ' per the MSRC advisory, then verify the resolved version in 'packages.lock.json' or by running 'dotnet list package | grep SemanticKernel'. If patching is blocked by change control, implement a YARA rule or WAF rule at the ingress layer to block prompt payloads containing path traversal sequences ('../', '%2e%2e%2f', '%252e') and command injection metacharacters (';', '|', '`', '$('). For a 2-person team without a WAF, a Kestrel middleware shim or an NGINX 'map' block with regex denial can serve as an immediate compensating control. Document the compensating control as a formal risk acceptance with a remediation deadline per NIST 800-61r3 §3.4 eradication guidance.

**Evidence:** Before applying the patch, snapshot: (1) the current 'packages.lock.json' and '.deps.json' from all affected application deployment directories to preserve the pre-patch version state for the incident record; (2) any web application firewall (WAF) or reverse proxy logs showing historical request patterns to AI agent endpoints — these may contain earlier exploitation attempts predating detection; (3) a hash inventory (SHA-256) of Semantic Kernel DLLs

currently on disk (e.g., 'Microsoft.SemanticKernel.dll') using 'Get-FileHash' or 'sha256sum' to confirm no tampered binaries were introduced via a prior compromise before the patch baseline is established.

**Step 4: Recovery — After patching, verify the installed Semantic Kernel version matches the remediated version listed in the Microsoft advisory. Re-enable any isolated services incrementally, monitoring for the same anomalous behaviors identified in the detection step. Confirm no unauthorized files were written or modified during any potential exploitation window by auditing file system changes on affected hosts.**

**NIST Phase:** Recovery

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

**Controls:** NIST SI-7 (Software, Firmware, and Information Integrity), NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

**Compensating:** Verify the patched version by running 'dotnet list package --include-transitive | grep SemanticKernel' in the application directory and confirming the version string matches the MSRC remediated release. For file system integrity verification of the exploitation window, use 'Get-ChildItem -Path -Recurse | Where-Object {$_.LastWriteTime -gt (Get-Date).AddDays(-30)} | Select-Object FullName,LastWriteTime | Export-Csv' (Windows) or 'find /app -newer /tmp/baseline_timestamp -type f' (Linux) to identify files written or modified during the exposure window. Re-enable services behind the host firewall rule incrementally — restore one endpoint at a time and monitor Sysmon Event ID 1 and 3 for 24 hours before widening access. Maintain Sysmon process creation logging on all Semantic Kernel hosts for a minimum 30-day post-recovery watch period.

**Evidence:** Before re-enabling isolated services, collect: (1) a post-patch file hash of 'Microsoft.SemanticKernel.dll' and all dependent assemblies to compare against the pre-patch baseline and confirm no unauthorized binary substitution occurred during the exposure window; (2) a filesystem diff report (using the CLI commands above) covering the period from earliest plausible exploitation to containment, focusing on the application root, temp directories ('C:\Windows\Temp', '/tmp', '/var/tmp'), and any directories the Semantic Kernel process had write access to — unexpected '.ps1', '.sh', '.py', or web shell files here are indicators of successful RCE exploitation; (3) final network connection logs from the recovering hosts to confirm no persistent C2 callback channels (e.g., reverse shells, scheduled outbound beaconing) are active before the firewall rules are lifted.

**Step 5: Post-Incident — Review your AI agent architecture for prompt injection attack surface broadly, not only in Semantic Kernel. Assess whether user-controlled input ever reaches framework-level execution without sanitization. Map this gap to NIST SP 800-53 SI-10 (Information Input Validation) and consider adding AI-specific threat modeling to your secure development lifecycle for any future AI agent projects.**

**NIST Phase:** Post Incident

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

**Controls:** NIST SI-10 (Information Input Validation), NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SA-11 (Developer Testing and Evaluation), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

**Compensating:** Conduct a 1-2 hour tabletop exercise specifically scoped to prompt injection pathways: trace every user-controlled input field in AI agent applications to the Semantic Kernel 'Kernel.InvokeAsync()', 'IChatCompletionService', or plugin execution surface and document whether sanitization occurs before that call. Use OWASP's LLM Top 10 (LLM01: Prompt Injection) as the threat model structure. Produce a data-flow diagram annotation showing trusted vs. untrusted input zones in each AI pipeline. Store incident artifacts (Sysmon logs, access logs, process snapshots, patch verification outputs) per NIST AU-11 (Audit Record Retention) requirements — a minimum 1-year retention is recommended for critical-severity incidents. File a lessons-learned report per NIST 800-61r3 §4 that specifically addresses whether input validation controls existed at Semantic Kernel plugin ingestion points and whether the vulnerability subscription process (NIST SI-5) captured this MSRC advisory before exploitation.

**Evidence:** For the post-incident record, preserve: (1) all detection-phase log exports (access logs, Sysmon EVTX files, network captures) with chain-of-custody documentation per NIST IR-9 (Information Spillage Response) standards, as these may be required for regulatory notification if PII or PHI was accessible to the Semantic Kernel process; (2) a copy of the application's Semantic Kernel integration code at the time of the incident (from version control — 'git log

--all --oneline' and 'git show ') to support root cause analysis of whether user input was passed unsanitized to framework execution calls; (3) the MSRC advisory text and the internal change record documenting patch application, to demonstrate remediation timeline for any compliance or audit review.

## Detection Guidance

Focus detection on two behavioral patterns specific to these CVEs. First, path traversal activity: look for HTTP request bodies or prompt API payloads containing directory traversal sequences ('../', '%2e%2e%2f', or URL-encoded equivalents) delivered to any Semantic Kernel agent endpoint. Second, unexpected code execution: monitor for anomalous child processes spawned by the Semantic Kernel host process, particularly shell interpreters (cmd.exe, powershell.exe, bash, sh) or scripting runtimes not expected in normal operation. In a SIEM, correlate process creation events (Sysmon Event ID 1 or Windows Security Event ID 4688) with the parent process of the Semantic Kernel service. For network-based detection, flag outbound connections initiated by the Semantic Kernel process to external IPs. No confirmed IOCs (hashes, IPs, domains) are available from this data set; IOC-based detection is not viable until exploitation activity is observed or threat intelligence is published. Behavioral detection is the primary viable approach at this stage.

## Framework Mappings

### MITRE-ATTACK

- **T1203** — Exploitation for Client Execution
- **T1190** — Exploit Public-Facing Application
- **T1059** — Command and Scripting Interpreter

### NIST-800-53R5

- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SI-10** — Information Input Validation
- **AC-3** — Access Enforcement

### OWASP-TOP10-2021

- **A03:2021** — Injection
- **A01:2021** — Broken Access Control

### CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **16.12** — Implement Code-Level Security Checks

### ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

### SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1203 | Exploitation for Client Execution | Execution |
| T1190 | Exploit Public-Facing Application | Initial-Access |
| T1059 | Command and Scripting Interpreter | Execution |

## Sources

| Source | URL | Tier |
|---|---|---|
| **gemini** | https://www.microsoft.com/en-us/security/blog/2026/05/07/when-promp... | T1 |
| **When prompts become shells: RCE vulnerabilities in AI agent ...** | https://www.microsoft.com/en-us/security/blog/2026/05/07/prompts-be... | T1 |
| **CVE-2026-25592 Detail - NVD** | https://nvd.nist.gov/vuln/detail/CVE-2026-25592 | T1 |
| **CVE-2026-26030 Detail - NVD** | https://nvd.nist.gov/vuln/detail/CVE-2026-26030 | T1 |
| **CVE-2026-25592: Semantic Kernel Path Traversal Flaw - SentinelOne** | https://www.sentinelone.com/vulnerability-database/cve-2026-25592/ | T3 |
| **NVD** | https://nvd.nist.gov/vuln/detail/CVE-2026-25592, CVE-2026-26030 | T1 |
| **Microsoft Security Advisory** | https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-2559... | T1 |

Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-05-08 14:01 UTC by TJS Security Command Center