INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-09 14:22 UTC

CVE-2026-42271: LiteLLM Command Injection Flaw Actively Exploited, Enables Unauthenticated RCE

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8

SCC Item ID	SCC-CVE-2026-0274
Type	CVE Vulnerability
CVE ID	CVE-2026-42271
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.6078 (98th percentile)
Affected Products	LiteLLM Proxy (specific version range not confirmed from available sources, verify against NVD/vendor advisory)
Published	11 hours ago
Discovery Source	Serper

Executive Summary

A critical unauthenticated remote code execution vulnerability in LiteLLM Proxy, a widely deployed open-source gateway used to route requests to large language model APIs, is reported by security researchers to be actively exploited. An attacker with network access to an exposed LiteLLM Proxy instance can execute arbitrary operating system commands without any credentials, gaining full control of the host. Organizations running LiteLLM Proxy in AI/ML pipelines, developer tooling, or production inference infrastructure face immediate risk of system compromise, data exfiltration, and lateral movement. Official NVD confirmation and vendor patch details are pending; verify against authoritative sources before deployment.

Technical Analysis

CVE-2026-42271 is a command injection vulnerability (CWE-78, CWE-77) in LiteLLM Proxy, an open-source SDK and reverse proxy for LLM API aggregation. According to Horizon3.ai third-party research, the vulnerability may chain with CVE-2026-48710 to achieve unauthenticated RCE; the second CVE likely bypasses an authentication or authorization check (CWE-306: Missing Authentication for Critical Function), providing the entry point that allows the command injection to be reached without credentials. The combined chain maps to

MITRE ATT&CK T1190 (Exploit Public-Facing Application), T1059 (Command and Scripting Interpreter), and T1203 (Exploitation for Client Execution). EPSS score of 0.608 places this vulnerability in the 98th percentile for exploitation probability. CVSS base score of 9.8 is from secondary sources; NVD has not yet published an official CVSS score. The qualitative rating of critical is supported by the attack vector (network, unauthenticated, no user interaction required) and impact (complete system compromise). Affected version range is unconfirmed - NVD entry (nvd.nist.gov/vuln/detail/CVE-2026-42271) and the official LiteLLM GitHub repository must be consulted for authoritative version and patch data. CISA KEV status is unconfirmed; verify directly at cisa.gov/known-exploited-vulnerabilities-catalog. Patch availability is unconfirmed from available sources.

Action Checklist

1. Step 1: Containment. Immediately audit all LiteLLM Proxy deployments for internet exposure. Restrict network access to LiteLLM Proxy admin and API interfaces to known internal IP ranges using firewall rules or security group policies; block all unauthenticated external access. If internet-facing exposure cannot be immediately removed, take the service offline until patched. Reference: NIST AC-4 (Information Flow Enforcement), enforce approved authorizations controlling information flow between connected systems; CIS 4.4 (Implement and Manage a Firewall on Servers).
2. Step 2: Detection. Review host-level process execution logs on all systems running LiteLLM Proxy for anomalous child processes spawned by the proxy service (e.g., bash, sh, cmd, PowerShell, curl, wget launched as subprocesses of the LiteLLM process). Check web server access logs for unusual POST requests to LiteLLM API endpoints, particularly those with shell metacharacters (semicolons, pipes, backticks, dollar signs) in parameter values. Query SIEM for outbound connections initiated by the LiteLLM process to external IPs, which may indicate post-exploitation C2 activity (T1059, T1190). Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs); D3-SFA (System File Analysis), monitor system logs and executables for modification. Specific IOCs are not confirmed from available sources; consult Horizon3.ai advisory and runZero blog for published indicators (URLs provided in sources require verification before operational use).
3. Step 3: Eradication. Apply the official LiteLLM patch once confirmed. Verify the patched version against the official LiteLLM GitHub repository (github.com/BerriAI/litellm) and NVD entry for CVE-2026-42271. Until a patch is applied, disable or remove LiteLLM Proxy from affected systems if the service is not operationally critical. On patched or restored systems, enforce authentication on all LiteLLM Proxy endpoints and remove any default or unused API key configurations. Reference: NIST SI-2 (Flaw Remediation); CIS 7.3 (Perform Automated Operating System Patch Management); CIS 7.4 (Perform Automated Application Patch Management); D3-CRO (Credential Rotation), rotate all API keys and credentials that were present on compromised or exposed LiteLLM hosts.
4. Step 4: Recovery. After patching, validate that no unauthorized accounts, scheduled tasks, cron jobs, or persistence mechanisms were created on affected hosts (MITRE T1059 post-exploitation follow-on). Re-enable LiteLLM Proxy only after confirming the patched version is running and authentication is enforced. Monitor process execution and outbound network connections from LiteLLM hosts for 72 hours post-restoration as a baseline verification period. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); NIST AC-2 (Account Management), review all accounts on affected hosts for unauthorized additions; D3-LAM (Local Account Monitoring), analyze local user accounts to detect unauthorized activity.
5. Step 5: Post-Incident. Document how LiteLLM Proxy entered the environment and whether it appeared in the enterprise asset inventory. If not inventoried, this represents a gap in asset visibility. Require all AI/ML infrastructure components, including LLM proxies, inference servers, and model API gateways, to

go through the same vulnerability management lifecycle as production application servers. Reference: NIST AC-6 (Least Privilege), review whether LiteLLM Proxy ran with more OS-level privilege than required; CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory); CIS 7.1 (Establish and Maintain a Vulnerability Management Process); D3-UAP (User Account Permissions), restrict service account permissions for AI/ML components.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and initiate breach notification review if forensic analysis confirms successful exploitation (evidence of child process execution from the LiteLLM process, unauthorized account creation, or data exfiltration), if LLM provider API keys exposed on the host were used to make unauthorized API calls, or if the affected host had access to sensitive internal data or systems beyond the LiteLLM service boundary — regulatory notification timelines (GDPR 72-hour, state breach laws) may be triggered depending on data classification of reachable systems.
Recovery Notes	Restore LiteLLM Proxy to production only after three conditions are verified in sequence: (1) pip show litellm confirms the patched version per NVD/GitHub advisory, (2) config.yaml enforces master_key authentication and all pre-incident LLM provider API keys have been rotated at the provider level, and (3) the persistence sweep (crontab, systemd units, authorized_keys, local accounts) returns clean. Maintain elevated process and network monitoring on restored LiteLLM hosts for a minimum of 72 hours post-restoration, reviewing auditd or Sysmon logs daily for anomalous child processes or outbound connections to non-LLM-provider destinations. If the host showed any confirmed exploitation indicators, treat it as fully compromised and reimage from a known-clean base rather than attempting in-place eradication.
Forensic Artifacts	LiteLLM Proxy access logs (location varies: ./logs/, /var/log/litellm/, or container stdout captured via `docker logs`) — search for POST requests to API endpoints containing shell metacharacters (;, |, `, $, &&) in JSON body parameters, which represent the command injection payload delivery vector for CVE-2026-42271 | Sysmon Event ID 1 (Process Create) or auditd execve records filtered to child processes of the LiteLLM Python/uvicorn process — attacker-executed OS commands (bash, curl, wget, python, nc) will appear as direct children of the LiteLLM process, providing the primary forensic signal of successful exploitation | SSH authorized_keys files for all users on affected hosts (~/.ssh/authorized_keys, /root/.ssh/authorized_keys) — post-RCE persistence via SSH key injection (MITRE T1098.004) is a common follow-on to unauthenticated command injection, and newly added keys establish the attacker's dwell timeline | LiteLLM config.yaml and associated environment variable files (.env) from the deployment directory — these contain all LLM provider API keys (OpenAI, Anthropic, Azure OpenAI, etc.) that were exposed on the compromised host and must be audited for unauthorized use via provider-side API usage logs | Files written to world-writable directories (/tmp, /var/tmp, /dev/shm) with timestamps falling within the exploitation window — command injection payloads commonly stage implants or download scripts to these paths; `find /tmp /var/tmp /dev/shm -type f -newer ` will surface attacker-dropped files before eradication

Per-Action IR Details

Step 1: Containment — Immediately audit all LiteLLM Proxy deployments for internet exposure. Restrict network access to LiteLLM Proxy admin and API interfaces to known internal IP ranges using firewall rules or

security group policies; block all unauthenticated external access. If internet-facing exposure cannot be immediately removed, take the service offline until patched. Reference: NIST AC-4 (Information Flow Enforcement) — enforce approved authorizations controlling information flow between connected systems; CIS 4.4 (Implement and Manage a Firewall on Servers).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: On Linux hosts: run `ss -tlnp | grep ` and `netstat -anp | grep LISTEN` to confirm which interfaces LiteLLM Proxy is bound to. Immediately apply an iptables rule to restrict inbound access: `iptables -A INPUT -p tcp --dport -s -j ACCEPT && iptables -A INPUT -p tcp --dport -j DROP`. On AWS/GCP/Azure, audit and tighten the security group or firewall rule attached to the LiteLLM host to deny 0.0.0.0/0 on the proxy port. For Docker-deployed instances, verify `docker ps --format '{{.Ports}}'` to identify any `0.0.0.0:` bindings and recreate the container with `--publish 127.0.0.1::` until patched.

Evidence: Before restricting network access, capture a full snapshot of current network exposure: output of `ss -tlnp`, `netstat -rn`, and `iptables -L -n -v` (Linux) or `netsh advfirewall show allprofiles` (Windows). Preserve cloud provider security group or VPC firewall rule configurations as-is (screenshot or API export) before any modification — these document the attack surface that existed during the exposure window. Capture active connection state with `ss -tnp state established` to identify any sessions already in progress to/from the LiteLLM process at time of containment.

Step 2: Detection — Review host-level process execution logs on all systems running LiteLLM Proxy for anomalous child processes spawned by the proxy service (e.g., bash, sh, cmd, PowerShell, curl, wget launched as subprocesses of the LiteLLM process). Check web server access logs for unusual POST requests to LiteLLM API endpoints, particularly those with shell metacharacters (semicolons, pipes, backticks, dollar signs) in parameter values. Query SIEM for outbound connections initiated by the LiteLLM process to external IPs, which may indicate post-exploitation C2 activity (T1059, T1190). Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs); D3-SFA (System File Analysis) — monitor system logs and executables for modification. Specific IOCs are not confirmed from available sources; consult Horizon3.ai advisory at horizon3.ai/attack-research/vulnerabilities/cve-2026-42271-chained-with-cve-2026-48710/ and runZero blog at runzero.com/blog/litellm/ for published indicators.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon (config with SwiftOnSecurity baseline) on LiteLLM hosts if not present; Event ID 1 (Process Create) will capture parent-child relationships — filter for the LiteLLM Python process (typically `python` or `uvicorn`) spawning `bash`, `sh`, `cmd.exe`, `curl`, or `wget`. On Linux without Sysmon, use auditd: `auditctl -a always,exit -F arch=b64 -S execve -F ppid=$(pgrep -f litellm) -k litellm_exec` to log all child process executions of the LiteLLM process. Parse LiteLLM access logs (default path: check deployment config or `~/.litellm/` and `/var/log/`) with: `grep -E '[;|`$(){}]' /path/to/litellm_access.log` to surface requests containing shell metacharacters in API parameters. Use Wireshark or `tcpdump -i any -w litellm_capture.pcap host ` to capture outbound connections from the proxy process for post-collection analysis.

Evidence: Preserve the following before any remediation: (1) LiteLLM Proxy access logs in full — default location varies by deployment but check `./logs/`, `/var/log/litellm/`, and container stdout/stderr via `docker logs > litellm_docker.log`; (2) Sysmon Event ID 1 or auditd `execve` records filtered to the LiteLLM process PID and its children, covering the full exposure window back to when the service was last known-clean; (3) `/proc//cmdline`, `/proc//environ`, and `/proc//fd/` for the live LiteLLM process if the host is not yet isolated — these capture runtime state including any injected environment variables or open file descriptors left by an attacker; (4) Shell history files (`~/.bash_history`, `/root/.bash_history`) on the host for commands run post-exploitation; (5) Network flow logs or `tcpdump` captures showing outbound connections from the LiteLLM host, specifically any connections to non-LLM-API destinations (i.e., not OpenAI, Anthropic, Azure OpenAI known IP ranges).

Step 3: Eradication — Apply the official LiteLLM patch once confirmed. Verify the patched version against the official LiteLLM GitHub repository (github.com/BerriAI/litellm) and NVD entry for CVE-2026-42271. Until a patch is applied, disable or remove LiteLLM Proxy from affected systems if the service is not operationally critical. On patched or restored systems, enforce authentication on all LiteLLM Proxy endpoints and remove any default or unused API key configurations. Reference: NIST SI-2 (no mapped control from the provided knowledge base for patch management — verify against your organization's SI family controls); CIS 7.3 (Perform Automated Operating System Patch Management); CIS 7.4 (Perform Automated Application Patch Management); D3-CRO (Credential Rotation) — rotate all API keys and credentials that were present on compromised or exposed LiteLLM hosts.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: 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: To verify the installed LiteLLM version: `pip show litellm | grep Version` or `pip3 show litellm | grep Version`; compare against the patched version listed in the GitHub release notes at github.com/BerriAI/litellm/releases. To enforce authentication without a commercial WAF, edit the LiteLLM `config.yaml` to set `general_settings.master_key` and ensure `general_settings.disable_spend_logs: false` so all API calls are logged. Rotate all LLM provider API keys (OpenAI, Anthropic, Azure, etc.) that were configured in the LiteLLM `config.yaml` or environment variables on the compromised host — treat all keys present on an exposed host as compromised regardless of confirmed exploitation. Use `find / -name 'config.yaml' -path '*/litellm/*' 2>/dev/null` to locate all LiteLLM config files and audit for hardcoded keys before redeployment.

Evidence: Before patching or reimaging: (1) Capture a full filesystem snapshot or disk image of affected hosts if active exploitation is confirmed or suspected — a command injection RCE at CVSS 9.8 with active exploitation warrants assuming full host compromise; (2) Export the LiteLLM `config.yaml` and any `.env` files present in the deployment directory — these contain all LLM provider API keys that must be considered compromised and rotated; (3) Collect `/etc/crontab`, `/etc/cron.d/`, `/var/spool/cron/crontabs/` and `systemctl list-units --type=service` output to document any persistence mechanisms installed post-exploitation before eradication removes them; (4) Run `find /tmp /var/tmp /dev/shm -type f -newer /proc/1/exe 2>/dev/null` to identify files written to world-writable directories by a post-exploitation payload.

Step 4: Recovery — After patching, validate that no unauthorized accounts, scheduled tasks, cron jobs, or persistence mechanisms were created on affected hosts (MITRE T1059 post-exploitation follow-on). Re-enable LiteLLM Proxy only after confirming the patched version is running and authentication is enforced. Monitor process execution and outbound network connections from LiteLLM hosts for 72 hours post-restoration as a baseline verification period. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); NIST AC-2 (Account Management) — review all accounts on affected hosts for unauthorized additions; D3-LAM (Local Account Monitoring) — analyze local user accounts to detect unauthorized activity.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AC-2 (Account Management)

Compensating: Pre-restoration persistence sweep (run as root on each affected host): `getent passwd | awk -F: '$3 >= 1000'` to identify non-system user accounts added post-compromise; `crontab -l -u ` and `cat /etc/cron.d/*` for attacker-planted cron jobs; `systemctl list-units --type=service --state=enabled` diffed against a known-good baseline for new or modified services; `find /etc/systemd/system /usr/lib/systemd/system -newer /tmp/pre_patch_timestamp -name '*.service'` for recently modified unit files. For 72-hour monitoring without SIEM, configure auditd rules to log all outbound connections from the LiteLLM process: `auditctl -a always,exit -F arch=b64 -S connect -F ppid=$(pgrep -f litellm) -k litellm_net` and review `/var/log/audit/audit.log` daily. Confirm patched version is running: `pip show litellm` on the restored host before re-enabling network access.

Evidence: Before re-enabling the LiteLLM service and restoring network access: (1) `/etc/passwd` and `/etc/shadow` diff against pre-incident baseline or a known-clean reference system to detect unauthorized local account creation; (2) Full output of `last`, `lastb`, and `who` to document all login activity during the incident window; (3) `/root/.ssh/authorized_keys` and all user `~/.ssh/authorized_keys` files — command injection RCE is commonly followed by SSH key persistence (MITRE T1098.004); (4) `ps auxf` and `/proc/*/cmdline` snapshot to confirm no attacker processes remain running before restoration; (5) Verification that LiteLLM `config.yaml` on the restored system contains only freshly rotated API keys and has `master_key` authentication enforced.

Step 5: Post-Incident — Document how LiteLLM Proxy entered the environment and whether it appeared in the enterprise asset inventory. If not inventoried, this represents a gap in asset visibility. Require all AI/ML infrastructure components — including LLM proxies, inference servers, and model API gateways — to go through the same vulnerability management lifecycle as production application servers. Reference: NIST AC-6 (Least Privilege) — review whether LiteLLM Proxy ran with more OS-level privilege than required; CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory); CIS 7.1 (Establish and Maintain a Vulnerability Management Process); D3-UAP (User Account Permissions) — restrict service account permissions for AI/ML components.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: To discover additional unregistered LiteLLM or LLM-proxy deployments across the environment: run `pip show litellm` via a remote execution tool (Ansible, SSH loop, or osquery `SELECT * FROM python_packages WHERE name='litellm'`) across all hosts in scope. Scan internal network ranges for LiteLLM's default port (4000/tcp) using `nmap -p 4000 --open -oN litellm_scan.txt` to surface shadow deployments not in the asset inventory. For least-privilege remediation: confirm the LiteLLM service account (or the user running the process) is a dedicated non-root service user — `ps aux | grep litellm` will show the running UID; if root, create a dedicated `litellm` system user and update the service unit file before redeployment. Document findings in the lessons-learned report and create a standing scan policy for AI/ML infrastructure ports.

Evidence: For the post-incident review record: (1) Full deployment audit output — how LiteLLM was installed (pip, Docker, Kubernetes), by whom, and when, sourced from package manager logs (`/var/log/dpkg.log`, `pip` install history, Docker image pull logs, or IaC repository commit history); (2) The service account UID/GID under which LiteLLM ran — if root or a high-privilege account, document as a contributing factor to blast radius; (3) Asset inventory records (or absence thereof) for the affected host(s) at time of incident — the gap between actual deployment and inventory registration is a key finding; (4) All LLM provider API key usage logs from OpenAI, Anthropic, Azure OpenAI, or other upstream providers for the compromised key period — submit key rotation requests and review provider-side usage logs for unauthorized API calls made using stolen keys during the exploitation window.

Detection Guidance

Primary detection focus is anomalous process execution on LiteLLM Proxy hosts. Look for child processes of the LiteLLM service spawning shell interpreters (bash, sh, cmd.exe, powershell.exe) or network utilities (curl, wget, nc, ncat). In web access logs, search for requests to LiteLLM API endpoints containing shell metacharacters in query parameters or request bodies: semicolons (;), pipe characters (|), backticks (`), dollar signs followed by parentheses ($(...)), and ampersands (&). Monitor for new user account creation, cron job modifications, or SSH authorized_keys changes on LiteLLM Proxy hosts following web traffic, indicators of post-exploitation persistence. Check for outbound connections to unusual external IPs or domains originating from the LiteLLM process, which may indicate reverse shell or C2 callback activity. Reference NIST AU-2 (Event Logging) to confirm that process execution and network connection logging is enabled on these hosts. Specific

confirmed IOCs (IPs, domains, file hashes) are not available from the sources analyzed. Before operational deployment, verify and resolve the Horizon3.ai advisory and runZero blog URLs listed in the action checklist to extract published indicators and proof-of-concept details that may inform detection rule development.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://nvd.nist.gov/vuln/detail/CVE-2026-42271`	NVD entry for CVE-2026-42271 — authoritative source for CVSS score, affected versions, and patch status; human verification required	HIGH
URL	`https://horizon3.ai/attack-research/vulnerabilities/cve-2026-42271-chained-with-cve-2026-48710/`	Horizon3.ai technical advisory describing the CVE-2026-42271 and CVE-2026-48710 exploit chain; may contain PoC details and network IOCs — URL not actively verified during this analysis	MEDIUM
URL	`https://www.runzero.com/blog/litellm/`	runZero blog on identifying exposed LiteLLM Proxy assets — may contain asset discovery queries; URL not actively verified during this analysis	MEDIUM

Framework Mappings

MITRE-ATTACK

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

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **IA-2** — Identification and Authentication (Organizational Users)
- **SI-10** — Information Input Validation
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A03:2021** — Injection

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **2.5** — Allowlist Authorized Software
- **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

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

Sources

Source	URL	Tier
	https://thehackernews.com/2026/06/litellm-flaw-cve-2026-42271-explo...	T3
Hackers are already exploiting a flaw in LiteLLM, a widely used ...	https://x.com/TheHackersNews/status/2064233564334698774	T3
CVE-2026-42271: LiteLLM Unauthenticated RCE | Horizon3.ai	https://horizon3.ai/attack-research/vulnerabilities/cve-2026-42271-...	T3
Active Exploitation of LiteLLM Flaw Enables Unauthenticated RCE ...	https://mallory.ai/stories/019eab36-e954-78da-94ba-af018a391b38	T3

Source	URL	Tier
LiteLLM Proxy vulnerabilities: How to find impacted assets - runZero	https://www.runzero.com/blog/litellm/	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-42271	T1