INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-11 05:54 UTC

Ollama Out-of-Bounds Read (Bleeding Llama): Critical Unauthenticated Memory Leak

CVE VULNERABILITY | HIGH | CVSS 8.6

SCC Item ID	SCC-CVE-2026-0153
Type	CVE Vulnerability
Severity	HIGH
CVSS Base Score	8.6
Affected Products	Ollama (version details unconfirmed, see CERT/CC VU#518910 and Cyera research for specifics)
Published	21 hours ago
Discovery Source	Serper

Executive Summary

A high-severity vulnerability dubbed 'Bleeding Llama' allows any unauthenticated attacker to remotely extract contents from the memory of an exposed Ollama AI inference server by sending a malformed file or API request. Any organization running Ollama with its API exposed to the internet is at immediate risk of having AI model weights, system prompts, or other in-memory data stolen without credentials or prior access. The business risk is highest for organizations whose AI deployments handle proprietary models, confidential system instructions, or sensitive data processed at inference time.

Technical Analysis

Ollama contains a high-severity out-of-bounds read vulnerability (CWE-125) in its handling of GGUF quantization files and API requests. An unauthenticated remote attacker can send a malformed GGUF file or crafted API request to trigger a memory read beyond the intended buffer boundary, leaking arbitrary process memory contents. Leaked memory may include model weights, system prompts, inference context, or other in-memory data depending on heap state at exploitation time. No authentication is required. CERT/CC has issued advisory VU#518910. A CVE identifier has not been confirmed from available sources; this item is tracked as a vulnerability advisory pending CVE publication. Affected versions are unconfirmed in available data; consult CERT/CC VU#518910 and Cyera's 'Bleeding Llama' research for specific version scope. MITRE techniques: T1190 (Exploit Public-Facing Application), T1552 (Unsecured Credentials). CWE-125 (Out-of-Bounds Read). CVSS base score: 8.6 (derived from researcher analysis; NVD CVSS score pending publication). No CISA KEV listing as of configuration date.

Action Checklist

1. Step 1: Containment, Immediately audit whether your Ollama API port (default: 11434) is exposed to the internet or untrusted networks. If exposed, restrict access via firewall or network ACL to trusted source IPs only. Do not rely on application-layer controls until a patch is confirmed applied. Reference CERT/CC VU#518910 for scope.
2. Step 2: Detection, Review web server and API access logs for Ollama on port 11434 for anomalous POST requests to inference or model load endpoints, particularly those referencing GGUF file handling or containing malformed payloads. Look for unexpected external source IPs in connection logs. No confirmed IOC signatures are available in current source data; behavioral anomalies in request volume or structure are the primary indicator.
3. Step 3: Eradication, Apply the vendor patch or mitigated version identified in CERT/CC VU#518910 and Cyera's research. Specific patch version details were not confirmed in available source data; consult https://kb.cert.org/vuls/id/518910 and https://www.cyera.com/research/bleeding-llama-critical-unauthenticated-memory-leak-in-ollama for authoritative remediation steps before acting. ■■ Vendor patch availability unconfirmed in current source data. If no patch is available from the Ollama project, implement network access controls as primary mitigation pending vendor release.
4. Step 4: Recovery, After patching, verify the updated Ollama version is running. Confirm the API is no longer accessible from untrusted networks. Monitor process memory usage and API access logs for anomalous patterns for at least 72 hours post-remediation. Rotate any sensitive system prompts or API keys that may have been in memory during the exposure window.
5. Step 5: Post-Incident, This vulnerability exposes a systemic control gap: AI inference services are frequently deployed without authentication or network segmentation. Implement mandatory authentication on all Ollama API endpoints. Enforce network-level access controls as a baseline requirement for all internal AI infrastructure. Review whether proprietary model weights or confidential system prompts exposed during the window require business notification or model retraining.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to legal, privacy, and senior leadership if Ollama API logs confirm external connections from untrusted IPs during the exposure window, particularly if the server hosted proprietary model weights, confidential system prompts containing PII or PHI, or API keys for downstream services — any confirmed data access may trigger trade secret, contractual breach notification, or regulatory (GDPR/HIPAA) obligations that exceed IR team authority.
Recovery Notes	After patching to the fixed Ollama version confirmed in CERT/CC VU#518910, validate the API is unreachable from untrusted networks via external port scan before resuming production inference workloads. Monitor Ollama API access logs and process memory metrics (via `top` or `ps` filtered on the ollama PID) for anomalous POST request patterns or memory growth for a minimum of 72 hours, as re-exploitation attempts against unpatched peer systems or delayed attacker callbacks are plausible. Any system prompt strings or API keys that were resident in Ollama process memory during the exposure window must be treated as compromised and rotated before the service is returned to full production use.

Forensic Artifacts	Ollama API access logs (~/.ollama/logs/server.log or journalctl -u ollama output): Primary evidence source for POST requests to /api/pull, /api/create, /api/push — captures attacker-controlled GGUF file submissions that trigger the out-of-bounds read, including source IPs and request sizes. | Network packet capture on TCP port 11434 (tcpdump/Wireshark): Captures the full HTTP request and response payload of exploitation attempts, including malformed GGUF content in request bodies and — critically — memory contents returned in the HTTP response body, which is the actual data leaked by the Bleeding Llama vulnerability. | Ollama process memory snapshot (gcore output or /proc//maps + /proc//mem): Documents what data was resident in the Ollama inference process address space during the exposure window, including loaded model weights, system prompt strings, and any secrets passed as environment variables — this defines the actual blast radius of the memory leak. | ~/.ollama/models/ directory listing with file hashes (sha256sum *.gguf): Establishes which GGUF model weight files were loaded and potentially readable from memory, supporting both the scope assessment of model weight exfiltration and any trade secret or IP notification decisions. | System environment variables and service unit files for the Ollama process (/proc//environ, /etc/systemd/system/ollama.service.d/): Reveals API keys, tokens, or configuration secrets that were in the process environment and therefore potentially accessible via the out-of-bounds memory read, scoping credential rotation requirements.

Per-Action IR Details

Step 1: Containment — Immediately audit whether your Ollama API port (default: 11434) is exposed to the internet or untrusted networks. If exposed, restrict access via firewall or network ACL to trusted source IPs only. Do not rely on application-layer controls until a patch is confirmed applied. Reference CERT/CC VU#518910 for scope.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 12.2 (Establish and Maintain a Secure Network Architecture)

Compensating: Run `ss -tlnp | grep 11434` (Linux) or `netstat -ano | findstr 11434` (Windows) to confirm Ollama is listening and on which interface. If exposed on 0.0.0.0, immediately apply an iptables rule: `iptables -I INPUT -p tcp --dport 11434 ! -s -j DROP`. On Windows, use `netsh advfirewall firewall add rule name='Block Ollama External' protocol=TCP dir=in localport=11434 action=block` then add a permit rule scoped to trusted IPs. Verify with `nmap -sV -p 11434 ` from an untrusted network to confirm the port is no longer reachable.

Evidence: Before restricting the port, capture current netstat state and active connections to port 11434 using `ss -tnp sport = :11434` to document any active or recent attacker sessions. Export `/var/log/syslog` or `/var/log/ollama/` (if present) to preserve pre-containment API connection history. On Linux hosts, capture `/proc/net/tcp` and `/proc/net/tcp6` to record raw socket state at time of containment. This preserves evidence of external IPs that had live connections to the Ollama API before the firewall rule was applied.

Step 2: Detection — Review web server and API access logs for Ollama on port 11434 for anomalous POST requests to inference or model load endpoints, particularly those referencing GGUF file handling or containing malformed payloads. Look for unexpected external source IPs in connection logs. No confirmed IOC signatures are available in current source data; behavioral anomalies in request volume or structure are the primary indicator.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST DE.AE-02 — Potentially adverse events are analyzed to better understand associated activities, CIS 8.2 (Collect Audit Logs)

Compensating: Query Ollama API access logs (default location: `~/.ollama/logs/` or journald via `journalctl -u ollama --since '7 days ago'`) for POST requests to `/api/pull`, `/api/create`, or `/api/push` originating from non-RFC1918 addresses. Use this bash one-liner to extract suspicious requests: `grep -E 'POST /(api/pull|api/create|api/push)' /var/log/ollama/server.log | awk '{print $1, $7, $NF}' | sort | uniq -c | sort -rn`. For network-level detection without a SIEM, run `tcpdump -i any -w /tmp/ollama_capture.pcap port 11434` and analyze with Wireshark, filtering on `tcp.port == 11434 && http.request.method == POST` to identify malformed or oversized GGUF-related payloads. Deploy the community Sigma rule targeting anomalous Ollama API POST volume if a log ingestion pipeline (e.g., Elastic Stack free tier) is available.

Evidence: Collect Ollama server logs from `~/.ollama/logs/server.log` or `journalctl -u ollama -o json` covering the full exposure window. Extract HTTP request sizes for POST calls — the Bleeding Llama out-of-bounds read is triggered by a malformed or oversized file upload (GGUF format), so unusually large Content-Length headers or multipart uploads to model endpoints are a key behavioral indicator. Also capture network flow data from the host's interface for port 11434 to identify source IPs, session duration, and data volume transferred outbound, which would reflect memory contents being leaked back to the attacker.

Step 3: Eradication — Apply the vendor patch or mitigated version identified in CERT/CC VU#518910 and Cyera's research. Specific patch version details were not confirmed in available source data — consult https://kb.cert.org/vuls/id/518910 and https://www.cyera.com/research/bleeding-llama-critical-unauthenticated-memory-leak-in-ollama for authoritative remediation steps before acting.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Before patching, snapshot the running Ollama process memory using `gcore $(pgrep ollama)` to preserve a forensic baseline of what was in memory at eradication time — this documents potential data exposure scope. Verify the installed Ollama version with `ollama --version` and compare against the fixed version published in CERT/CC VU#518910. After applying the patch via `curl -fsSL https://ollama.com/install.sh | sh` (or your package manager), confirm the new binary hash matches the vendor-published checksum: `sha256sum $(which ollama)`. Restart the service with `systemctl restart ollama` and re-verify version. If patching is not immediately possible, enforce the network ACL from Step 1 as the sole compensating control and document the exception per NIST SI-2 requirements.

Evidence: Before applying the patch, capture the output of `ollama --version`, the SHA-256 hash of the current Ollama binary (`sha256sum $(which ollama)`), and a full process listing (`ps auxf | grep ollama`) to document the pre-patch state for the incident record. If the system was actively exploited, use `strings /proc/$(pgrep ollama)/mem` cautiously (or `gcore` as above) to assess what in-memory data — model weights, system prompt strings, API key fragments — may have been accessible to an attacker exploiting the out-of-bounds read. Preserve this evidence before the patch clears the process memory state.

Step 4: Recovery — After patching, verify the updated Ollama version is running. Confirm the API is no longer accessible from untrusted networks. Monitor process memory usage and API access logs for anomalous patterns for at least 72 hours post-remediation. Rotate any sensitive system prompts or API keys that may have been in memory during the exposure window.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IA-5 (Authenticator Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.2 (Use Unique Passwords)

Compensating: Verify patched version with `ollama --version` and confirm network isolation with `nmap -sV -p 11434 ` from an external vantage point. For credential rotation: enumerate all API keys, system prompt files, and model configuration files that were loaded into Ollama process memory during the exposure window by reviewing `~/.ollama/models/` and any custom Modelfile paths (`find / -name 'Modelfile' 2>/dev/null`). Revoke and reissue any

downstream API keys (e.g., OpenAI-compatible API keys passed as environment variables) using `grep -r 'API_KEY\|SECRET\|TOKEN' /etc/systemd/system/ollama.service.d/` to identify what secrets were in the process environment. Set up a lightweight 72-hour monitoring cron job: `*/15 * * * * ss -tnp sport = :11434 | grep -v '127.0.0.1\|' >> /var/log/ollama_external_conns.log` to alert on any resumed external connections.

Evidence: Document the post-patch Ollama binary hash and version as the clean-state baseline. Review Ollama API logs for the 72-hour post-patch window to confirm absence of malformed POST requests to `/api/pull` or `/api/create`. Catalog all system prompt strings and model configuration files (`Modelfile` contents) that were resident in memory during the exposure window — these constitute the data loss scope and must be documented for the post-incident report and any required business notification decisions.

Step 5: Post-Incident — This vulnerability exposes a systemic control gap: AI inference services are frequently deployed without authentication or network segmentation. Implement mandatory authentication on all Ollama API endpoints. Enforce network-level access controls as a baseline requirement for all internal AI infrastructure. Review whether proprietary model weights or confidential system prompts exposed during the window require business notification or model retraining.

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 IA-3 (Device Identification and Authentication), NIST AC-3 (Access Enforcement), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Implement Ollama API authentication immediately using a reverse proxy: deploy Nginx or Caddy (both free) in front of port 11434 with HTTP Basic Auth or a bearer token requirement, blocking all direct external access to the raw Ollama port. Example Nginx config snippet: `location / { auth_basic 'Ollama API'; auth_basic_user_file /etc/nginx/.htpasswd; proxy_pass http://127.0.0.1:11434; }`. For model weight and system prompt inventory, run `find ~/.ollama/models/ -type f -name '*.gguf' | xargs sha256sum > /var/log/ollama_model_inventory_$(date +%Y%m%d).txt` to establish a post-incident model integrity baseline. If proprietary model weights were exposed, consult legal/business stakeholders about trade secret implications — this is an escalation trigger, not a technical decision.

Evidence: Compile the full exposure window timeline: first external connection to port 11434 from logs, patch application timestamp, and network isolation timestamp. Document all GGUF model files resident in `~/.ollama/models/` during the exposure window (names, sizes, hashes) to define the potential model weight exfiltration scope. Preserve all Ollama API access logs from the exposure window in write-protected, integrity-verified storage (use `sha256sum` on archived logs) per NIST AU-9 (Protection of Audit Information) requirements for post-incident review and potential legal hold.

Detection Guidance

Monitor Ollama API access logs (default port 11434) for POST requests to model load or inference endpoints originating from unexpected external IPs. Flag requests containing GGUF file references or abnormally large or malformed request bodies. No confirmed exploit signatures or specific IOC patterns are available in current source data; treat any unauthenticated external access to the Ollama API as a high-priority alert given this vulnerability. If network flow logging is in place, look for unexpectedly large outbound response sizes from the Ollama process, which may indicate memory content being returned to the caller. SIEM rule suggestion: alert on connections to port 11434 from non-whitelisted source IPs, especially where response bytes exceed a defined threshold for normal inference traffic.

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1552** — Unsecured Credentials

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

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures

ISO-27001-2022

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1552	Unsecured Credentials	Credential-Access

Sources

Source	URL	Tier
	https://thehackernews.com/2026/05/ollama-out-of-bounds-read-vulnera...	T3
Ollama Out-of-Bounds Read Vulnerability Allows Remote Process ...	https://www.reddit.com/r/cybersecurity/comments/1t96jg8/ollama_outo...	T3
Bleeding Llama: Critical Unauthenticated Memory Leak in Ollama	https://www.cyera.com/research/bleeding-llama-critical-unauthentica...	T3
Ollama Out-of-Bounds Read Vulnerability Allows Remote Process ...	https://x.com/TheCyberSecHub/status/2053456826382115217	T3

Source	URL	Tier
VU#518910 - Ollama GGUF Quantization Remote Memory Leak	https://kb.cert.org/vuls/id/518910	**T3**

Generated 2026-05-11 05:54 UTC by TJS Security Command Center