INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-18 06:15 UTC

Prompt Layer Becomes a Live Attack Surface: What Security Teams Must Do Before AI Workloads Reach Production

SECURITY ANALYSIS | MEDIUM | CVSS 5.0

SCC Item ID	SCC-STY-2026-0142
Type	Security Analysis
Severity	MEDIUM
CVSS Base Score	5.0
Affected Products	CrowdStrike Falcon AIDR, CrowdStrike Falcon Cloud Security, Falcon Container Sensor, Falcon Next-Gen SIEM, OpenAI-compatible LLM clients running in Kubernetes environments
Discovery Source	Rss:T1 Threatintel

Executive Summary

AI workloads deployed in Kubernetes environments have introduced an attack surface that most enterprise security stacks cannot see: the prompt layer. Attackers can inject malicious instructions directly into large language model inputs, bypassing signature-based and behavioral detection tools entirely, with no CVE to patch and no network anomaly to catch. This represents a structural visibility gap, not a discrete software defect, and organizations scaling AI applications without prompt-layer instrumentation are operating blind to a threat class OWASP has ranked the top risk in LLM deployments.

Technical Analysis

Prompt injection exploits a fundamental property of large language models: the model cannot reliably distinguish between legitimate user input and adversarial instructions embedded within that input. This is not a software bug with a patch; it is an architectural characteristic of how LLMs process natural language. OWASP classifies it as LLM01, the top risk in LLM applications, and CrowdStrike has reported tracking a large and growing number of prompt injection techniques, a figure that signals rapid adversarial maturation rather than experimental activity.

The attack surface is particularly acute in Kubernetes-hosted AI workloads, where containerized LLM applications often interact with internal APIs, data stores, and orchestration systems. An attacker who successfully injects instructions into a model's prompt context can potentially redirect the model to exfiltrate data (mapping to T1602, T1552), execute commands within the application's permission scope (T1059, T1203), manipulate outputs delivered downstream (T1565), or abuse valid service accounts and tokens the container already holds (T1078, T1610, T1190). The attack chain does not require malicious code; the payload is natural

language.

The defensive gap is instrumentation, not capability. Existing EDR and network detection tools were designed to identify anomalous binaries, lateral movement, and malicious traffic patterns. They have no visibility into what passes between a user interface and an LLM inference endpoint. Without prompt-layer telemetry, security teams cannot log what instructions the model received, cannot detect injection patterns, and cannot correlate model behavior with downstream system actions. CrowdStrike's Falcon AIDR capability addresses this by instrumenting the prompt layer within Kubernetes-hosted AI workloads, integrating telemetry into Falcon Cloud Security, Falcon Container Sensor, and Falcon Next-Gen SIEM. This approach treats prompt interactions as a loggable, monitorable event stream rather than an opaque application function.

The absence of a CVE assignment is significant for security teams to internalize. Vulnerability management workflows built around CVE triage, EPSS scores, and KEV deadlines will not surface this risk. It must be addressed through threat modeling, architecture review, and runtime instrumentation, not patch prioritization.

Action Checklist

1. Step 1: Assess exposure, inventory every AI application in your environment, specifically any LLM-integrated workload running in Kubernetes; determine whether prompt inputs from users or external systems reach the model without validation or logging
2. Step 2: Review controls, verify whether your current EDR, CSPM, or SIEM stack has any visibility into prompt-layer interactions; if not, identify the instrumentation gap and evaluate runtime AI detection tooling such as CrowdStrike Falcon AIDR, open-source instrumentation frameworks, or equivalent vendor capabilities
3. Step 3: Update threat model, add prompt injection (OWASP LLM01) as a named threat pattern in your threat register; map it to MITRE ATT&CK techniques T1059, T1203, T1552, T1565, T1602, and T1078 to align detection engineering and red team planning
4. Step 4: Audit container permissions, review service account scopes, API permissions, and data access rights granted to Kubernetes pods running AI workloads; a prompt injection attack operates within whatever permissions the container already holds, so least-privilege enforcement directly limits blast radius
5. Step 5: Communicate findings, brief leadership on the visibility gap using concrete terms: current security tooling cannot log or detect malicious instructions sent to AI applications; frame the investment in prompt-layer monitoring as closing a known blind spot, not as speculative AI risk

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal/privacy counsel if any prompt-layer audit (once instrumentation is deployed) reveals exfiltration of PII, PHI, or credentials from AI workload responses, or if a Kubernetes service account with secrets access shows API calls to sensitive namespaces following external prompt submissions — both conditions may trigger breach notification obligations under GDPR, HIPAA, or state privacy laws regardless of whether a traditional CVE-based exploit was involved.

Recovery Notes	Once prompt-layer instrumentation is deployed and RBAC is scoped to least privilege, establish a 30-day enhanced monitoring period during which all LLM input/output logs are reviewed daily for anomalous instruction patterns, unexpected tool calls, or responses containing internal system information. Verify recovery completeness by re-running the `kubectl auth can-i` permission audit against all AI workload service accounts post-remediation and confirming Falcon Cloud Security or kube-bench shows no remaining high-severity RBAC findings. Long-term, integrate prompt injection test cases (using open-source tools such as garak or promptbench) into the CI/CD pipeline for all LLM-integrated workloads before they reach production, closing the structural gap this advisory identifies.
Forensic Artifacts	Kubernetes API server audit logs (`/var/log/kube-apiserver-audit.log` or cloud-provider equivalent): filter for `verb: create` or `verb: get` on `resource: secrets` by AI workload service accounts — a prompt injection payload instructing the LLM to retrieve credentials would surface here as anomalous secrets access with no human initiator | LLM inference server stdout/stderr pod logs via `kubectl logs --timestamps=true`: if OpenAI-compatible endpoints are in use, request bodies logged at debug level will contain raw prompt text including any injected instructions — absence of these logs is itself a forensic finding confirming the visibility gap | Kubernetes NetworkPolicy and Envoy/Istio access logs for AI workload namespaces: a successful prompt injection leading to SSRF or data exfiltration would produce outbound connections from the inference pod to unexpected internal cluster IPs or external endpoints not in the pod's normal traffic baseline | Falcon Container Sensor process tree telemetry (if deployed): look for child processes spawned by the inference server process (e.g., `python`, `uvicorn`, `gunicorn`) — prompt injection triggering tool-use or code execution via LangChain/LlamaIndex agent frameworks would appear as anomalous subprocess creation under the model server PID | ConfigMap and Secret access records in Kubernetes audit logs: LLM agent frameworks with cluster API access (e.g., LangChain Kubernetes toolkit) that are compromised via prompt injection would show `get` or `list` operations on ConfigMaps and Secrets from the AI pod's service account in namespaces the workload has no legitimate reason to access

Per-Action IR Details

Step 1: Assess exposure — inventory every AI application in your environment, specifically any LLM-integrated workload running in Kubernetes; determine whether prompt inputs from users or external systems reach the model without validation or logging

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establishing IR capability and asset visibility before incidents occur

Controls: NIST IR-4 (Incident Handling) — requires preparation including understanding of assets subject to incident scope, NIST SI-4 (System Monitoring) — requires monitoring capability for system components; cannot monitor what is not inventoried, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — inventory must include AI/LLM workloads as enterprise assets with potential to store or process sensitive data, CIS 2.1 (Establish and Maintain a Software Inventory) — LLM runtimes, inference servers (e.g., vLLM, Ollama, OpenAI-compatible endpoints), and orchestration frameworks (LangChain, LlamaIndex) must appear in software inventory

Compensating: Run `kubectl get pods --all-namespaces -o json | jq '.items[] | select(.spec.containers[].env[]?.value | test("openai|anthropic|llm|gpt|ollama|vllm"; "i")) | {namespace: .metadata.namespace, pod: .metadata.name}'` to surface pods with LLM-related environment variables. Supplement with `kubectl get ingress --all-namespaces` to identify externally reachable AI endpoints. For non-Kubernetes deployments, use `ss -tlnp` or `netstat -tlnp` on inference hosts to enumerate listening ports, cross-referenced against process names.

Evidence: Before beginning inventory, snapshot current state: capture `kubectl get pods,services,ingress,configmaps,secrets --all-namespaces -o yaml > k8s_ai_inventory_snapshot.yaml` as a

timestamped baseline. Preserve any existing API gateway or reverse proxy access logs (e.g., NGINX `/var/log/nginx/access.log`, Istio Envoy access logs via `kubectl logs -n istio-system`) that may show historical prompt submission patterns to LLM endpoints — these establish pre-incident traffic baselines for later comparison.

Step 2: Review controls — verify whether your current EDR, CSPM, or SIEM stack has any visibility into prompt-layer interactions; if not, identify the instrumentation gap and evaluate runtime AI detection tooling such as CrowdStrike Falcon AIDR or equivalent capabilities

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Identifying visibility gaps in monitoring capability that prevent adverse event detection

Controls: NIST SI-4 (System Monitoring) — requires monitoring of system components for attacks and indicators of potential attacks; prompt-layer interactions are system inputs that must fall within monitoring scope, NIST AU-2 (Event Logging) — requires identifying event types the system is capable of logging; prompt input/output is a loggable event type absent from most current SIEM configurations, NIST AU-12 (Audit Record Generation) — audit records must be generated for defined auditable events; LLM prompt submissions and model responses are auditable events not captured by signature-based EDR, CIS 8.2 (Collect Audit Logs) — audit log collection must be enabled across enterprise assets; AI inference endpoints are enterprise assets whose prompt-layer events are not logged by default in Kubernetes

Compensating: Without Falcon AIDR or commercial AI security tooling, deploy an OpenTelemetry-instrumented sidecar proxy in each AI workload pod to log prompt inputs and model outputs to a local file or syslog sink. Use the LiteLLM proxy (open source) as an OpenAI-compatible intercepting layer that logs full request/response bodies to stdout, which Kubernetes captures in pod logs accessible via `kubectl logs`. Write a Sigma rule targeting HTTP POST bodies to `/v1/chat/completions` or `/v1/completions` endpoints containing known prompt injection patterns (e.g., `ignore previous instructions`, `DAN`, `system:`, `[INST]`) using Sigma's http_body field for log sources that capture request payloads.

Evidence: Capture the current absence of evidence as evidence itself: export your SIEM's existing detection rule inventory and confirm zero rules targeting LLM API endpoint paths (`/v1/chat/completions`, `/v1/completions`, `/v1/messages`). Pull Kubernetes pod logs for all AI workload pods via `kubectl logs --since=720h > pod_ai_logs.txt` — the absence of prompt content in these logs confirms the instrumentation gap. If Falcon Container Sensor is deployed, verify in Falcon console whether process tree telemetry shows the inference server process receiving external HTTP connections with payload inspection enabled.

Step 3: Update threat model — add prompt injection (OWASP LLM01) as a named threat pattern in your threat register; map it to MITRE ATT&CK techniques T1059, T1203, T1552, T1565, T1602, and T1078 to align detection engineering and red team planning

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Maintaining threat intelligence and updating IR capability to reflect current threat landscape

Controls: NIST IR-8 (Incident Response Plan) — IR plan must be updated to reflect current threat patterns; prompt injection as OWASP LLM01 is an unaddressed threat pattern in most existing plans, NIST RA-3 (Risk Assessment) — risk assessment must include identification of threats to organizational systems; LLM-integrated Kubernetes workloads represent a new threat surface requiring explicit threat register entry, NIST SI-5 (Security Alerts, Advisories, and Directives) — requires receiving and acting on security advisories; OWASP LLM Top 10 and MITRE ATLAS framework entries constitute authoritative advisories for this threat class, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — vulnerability management process must account for vulnerability classes without CVEs, including architectural weaknesses like unvalidated prompt inputs

Compensating: Document the threat model update using the MITRE ATT&CK Navigator (free, browser-based at attack.mitre.org/resources/attack-navigator/) — create a layer file marking T1059 (Command and Scripting Interpreter), T1203 (Exploitation for Client Execution), T1552 (Unsecured Credentials), T1565 (Data Manipulation), T1602 (Data from Configuration Repository), and T1078 (Valid Accounts) with a custom tactic label 'Prompt Injection Initial Access/Execution.' Export the layer JSON as a versioned artifact in your threat register. For the OWASP LLM01 entry,

reference the OWASP LLM AI Security Top 10 v1.1 directly at owasp.org/www-project-top-10-for-large-language-model-applications/ — note this URL should be validated at time of access.

Evidence: Before finalizing ATT&CK mappings, collect any existing red team reports, penetration test findings, or bug bounty submissions that touched AI workloads — these may already contain prompt injection attempts that were not classified as such. Pull Falcon Next-Gen SIEM (if deployed) query history to determine whether any analyst has previously searched for patterns consistent with T1059 originating from inference server processes, establishing whether the gap is in detection rules or in analyst awareness.

Step 4: Audit container permissions — review service account scopes, API permissions, and data access rights granted to Kubernetes pods running AI workloads; a prompt injection attack operates within whatever permissions the container already holds, so least-privilege enforcement directly limits blast radius

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment: Implementing controls that limit the scope and impact of exploitation before or during an active incident

Controls: NIST AC-6 (Least Privilege) — systems must operate with the minimum permissions necessary; Kubernetes service accounts for LLM pods must not hold cluster-admin, secrets-read, or broad API server permissions that a prompt injection payload could leverage, NIST IR-4 (Incident Handling) — containment strategies must be defined and implementable; reducing service account scope is a pre-containment action that limits what an attacker controlling the LLM process can do, CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts) — AI workload service accounts must not hold administrative privileges; a compromised LLM pod with cluster-admin binds T1078 (Valid Accounts) to near-complete cluster compromise, CIS 4.6 (Securely Manage Enterprise Assets and Software) — Kubernetes RBAC configurations must be version-controlled and reviewed; service account permissions for AI pods should be declared in manifests and audited against least-privilege baselines

Compensating: Run `kubectl auth can-i --list --as=system:serviceaccount::` for each AI workload service account to enumerate effective permissions. Use `kubectl get rolebindings,clusterrolebindings --all-namespaces -o json | jq '.items[] | select(.subjects[]?.name | test(""))' ` to surface all bindings. For secrets access specifically: `kubectl get clusterrolebinding -o json | jq '.items[] | select(.roleRef.name == "cluster-admin") | .subjects'` — any AI workload service account appearing here is an immediate remediation priority. Use kube-bench (free, CIS Kubernetes Benchmark tool) to validate broader RBAC posture: `docker run --rm --pid=host -v /etc:/etc:ro -v /var:/var:ro aquasec/kube-bench`.

Evidence: Before modifying any permissions, export the current RBAC state as a forensic baseline: `kubectl get rolebindings,clusterrolebindings,serviceaccounts,roles,clusterroles --all-namespaces -o yaml > rbac_baseline_$(date +%Y%m%d).yaml`. If Falcon Cloud Security (CSPM) is deployed, export the current misconfigurations report for the Kubernetes cluster — specifically findings in the 'Excessive Permissions' and 'Privileged Containers' categories — to document pre-remediation exposure. Check Kubernetes API server audit logs (typically at `/var/log/kube-apiserver-audit.log` or via cloud provider logging) for any `create`, `get`, or `list` actions on `secrets` resources by AI workload service accounts in the past 30 days.

Step 5: Communicate findings — brief leadership on the visibility gap using concrete terms: current security tooling cannot log or detect malicious instructions sent to AI applications; frame the investment in prompt-layer monitoring as closing a known blind spot, not as speculative AI risk

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Lessons learned, capability gap documentation, and communicating findings to improve organizational security posture

Controls: NIST IR-6 (Incident Reporting) — requires reporting of incidents and security findings to appropriate organizational personnel; visibility gap findings constitute a reportable security posture deficiency requiring leadership awareness, NIST IR-8 (Incident Response Plan) — IR plan must include communication procedures; the prompt-layer blind spot must be documented as a named gap in the plan's coverage scope with a remediation timeline, NIST IR-2 (Incident Response Training) — training must be updated to reflect current threats; leadership briefing is the first step in ensuring organizational awareness that prompt injection is an IR-relevant threat, not solely a development concern, CIS 7.2 (Establish and Maintain a Remediation Process) — remediation process must include risk-based prioritization;

framing prompt-layer monitoring as closing a known blind spot positions it within the existing remediation process rather than as a net-new budget category

Compensating: For teams without a formal GRC platform, document the gap using a one-page risk register entry: asset (LLM workloads in Kubernetes), threat (prompt injection / OWASP LLM01), current control state (no prompt-layer logging or detection), residual risk (attacker can issue arbitrary instructions to AI systems with no forensic trail), and proposed control (OpenTelemetry sidecar logging + Sigma rules as interim; Falcon AIDR or equivalent as target state). Attach the `kubectl auth can-i` output from Step 4 and the SIEM rule gap documentation from Step 2 as supporting evidence — concrete data converts a theoretical risk conversation into a documented findings briefing.

Evidence: Compile the evidence package from all prior steps before the leadership briefing: the K8s inventory snapshot from Step 1, the SIEM rule gap export from Step 2, the ATT&CK Navigator layer from Step 3, and the RBAC baseline YAML from Step 4. This package serves as the post-incident documentation artifact required by NIST IR-5 (Incident Monitoring) — even in the absence of a confirmed incident, documenting a known visibility gap and its remediation trajectory is an auditable record that demonstrates due diligence to regulators and auditors.

Detection Guidance

Because prompt injection leaves no malicious binary and no anomalous network signature, detection depends on logging and monitoring what the model receives and what it causes downstream.

Prompt-layer telemetry: If you have runtime instrumentation (Falcon AIDR or equivalent), alert on inputs containing instruction-override patterns, role-reassignment language, delimiter manipulation, or base64-encoded payloads embedded in natural language requests. Establish a baseline of normal prompt structure for your application and treat significant deviations as a detection signal.

Application behavior anomalies: Monitor for LLM-integrated applications making unexpected API calls, accessing data stores outside their normal scope, or generating outputs that include internal system references. These downstream behaviors may be the only observable artifact of a successful injection in environments without prompt-layer logging.

Kubernetes audit logs: Review pod-level API activity for service account token usage that deviates from baseline, unexpected exec commands into containers (T1059), or container deployments initiated by workloads that should not have that permission (T1610). A compromised AI workload will typically attempt to move laterally using the permissions it already holds.

Log gap audit: As a foundational step, verify that your SIEM receives logs from every layer of the AI application stack, including inference endpoints, model API gateways, and any retrieval-augmented generation (RAG) data connectors. Absence of logs from these components is itself a detection gap that should be flagged.

Framework Mappings

MITRE-ATTACK

- **T1203** — Exploitation for Client Execution
- **T1078** — Valid Accounts
- **T1602** — Data from Configuration Repository
- **T1610** — Deploy Container
- **T1190** — Exploit Public-Facing Application
- **T1552** — Unsecured Credentials

- **T1565** — Data Manipulation
- **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
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **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
- **SC-28** — Protection of Information at Rest
- **AT-2** — Literacy Training and Awareness

OWASP-TOP10-2021

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

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.26** — Application security requirements
- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored
- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1203	Exploitation for Client Execution	Execution
T1078	Valid Accounts	Defense-Evasion
T1602	Data from Configuration Repository	Collection
T1610	Deploy Container	Defense-Evasion
T1190	Exploit Public-Facing Application	Initial-Access
T1552	Unsecured Credentials	Credential-Access
T1565	Data Manipulation	Impact
T1059	Command and Scripting Interpreter	Execution

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/falcon-aidr-detects-threats-...	T3
	https://www.crowdstrike.com/en-us/blog/falcon-aidr-detects-threats-...	T3
	https://www.crowdstrike.com/en-us/blog/crowdstrike-expands-real-tim...	T3
	https://www.crowdstrike.com/en-us/blog/crowdstrike-secures-growing-...	T3
CrowdStrike Extends Falcon AIDR to Kubernetes AI Workloads	https://techjacksolutions.com/scc-intel/crowdstrike-extends-falcon-...	T3

Generated 2026-05-18 06:15 UTC by TJS Security Command Center