elastic | The Search AI Company

How Elastic's Search AI Platform supports the OWASP Top 10 for LLM Applications with LLM security and observability

Mitigating the risks identified in the OWASP Top 10 for LLM Applications requires a unified platform approach. Elastic's Search AI Platform provides this holistic solution by combining full lifecycle observability with powerful security analytics and a ***Zero Trust Architecture***.

Elastic's mitigation strategy is to provide complete, ***end-to-end*** visibility across the entire large language model (LLM) application stack. By ingesting and normalizing logs, metrics, and traces, Elastic creates a single, correlated data store. Upon this foundation, ***Elastic Security*** and ***Elastic Observability*** work in tandem. Machine learning can detect anomalous behavior and correlate suspicious LLM activity and its downstream effects, transforming low-confidence signals into actionable alerts. This unified approach allows organizations to address the full spectrum of modern AI threats and attacks.

In this white paper, we'll look at the OWASP issues one by one, detailing how Elastic can mitigate risk and reduce vulnerability.

LLM01:2025 Prompt Injection

Vulnerability deep dive

Prompt Injection occurs when a user's input manipulates the model's behavior in unintended ways.[1] This manipulation can bypass safety guidelines, enable unauthorized access, or influence critical decisions. The attack vector can be Direct, Indirect, or Multi-Modal injection.

End-to-end mitigation with Elastic

Layer 1: Comprehensive telemetry and auditing

The first step is to log every prompt, its corresponding output, and the rich context surrounding the interaction.[2]

Automated log collection: Elastic's integrations for leading large language model (LLM) providers automatically collect detailed model invocation logs and metrics. These integrations capture prompts, responses, metadata about tool calls, parameters, and returned data, which is essential for detecting abuse.[3]

Standardized data with ECS: This telemetry is mapped to the new gen_ai fields within the Elastic Common Schema (ECS).

Custom application instrumentation: For bespoke LLM applications, Elastic application performance monitoring (APM) or the OpenTelemetry integration can be used to capture and forward all LLM interaction data, ensuring no blind spots exist in the application stack.[4]

Layer 2: Behavioral and anomaly detection

Log rate and pattern analysis: Anomaly detection jobs can be configured to monitor the rate and patterns of LLM logs. A sudden spike in logs containing specific keywords (e.g., "ignore your instructions"), or a change in the statistical properties of prompts from a particular user, can be early indicators of an attack.[5] The system can then automatically surface deviations from the set baseline that may indicate a prompt injection attack in progress.[6]

Layer 3: Correlation and downstream effect detection

A successful injection is frequently invisible until it triggers a malicious action. Elastic's Search AI Platform can correlate events across the entire technology stack, providing a unique and decisive advantage. Elastic Security monitors the backend systems for the malicious effect.

Layer 4: Integration and red teaming

While Elastic provides world-class detection and response, a defense-in-depth strategy also includes preinference prevention. Elastic can be integrated with LLM security gateways or firewalls from partners like Prompt Security or open source tools like Rebuff and LLM-Guard. These tools analyze and block malicious prompts *before* they reach the model. All logs and alerts from these third-party prevention tools should be ingested into Elastic. This makes the Search AI Platform the single pane of glass for all generative AI (GenAI) security events, correlating prevention-based alerts with detection-based findings from the rest of the stack.[7]

LLM02:2025 Sensitive Information Disclosure

Vulnerability deep dive

Sensitive Information Disclosure occurs when an LLM-based application exposes confidential data, such as personally identifiable information (PII), financial details, proprietary algorithms, or security credentials, through its output. This exposure can result from two primary failure modes: the model regurgitating sensitive data it memorized from its training set, or a user inputting sensitive information into a prompt, which is then logged insecurely or inadvertently revealed to other users. A notable historical parallel is the "Proof Pudding" attack, where disclosed training data enabled model inversion attacks, allowing adversaries to reconstruct sensitive inputs.[1]

The Elastic mitigation strategy: Data-centric security

Layer 1: Proactive data sanitization and filtering

Elastic offers concrete, out-of-the-box capabilities to implement data sanitization both in transit and at rest, creating a powerful, multistage defense against accidental data exposure.

Sanitization in transit: Elastic APM agents can be configured with built-in and custom data filters to automatically scrub, mask, or obfuscate sensitive data from HTTP headers and bodies before they are sent to the APM server for monitoring.[8] This is a critical proactive control for LLM applications, as it can prevent user-submitted PII or credentials in prompts from ever being logged.

Sanitization at rest: For data being ingested into Elasticsearch, such as documents for a RAG knowledge base, **ingest pipelines** provide a server-side mechanism for sanitization. Using

processors like grok, dissect, or script, administrators can define patterns to identify and redact sensitive information like social security numbers, credit card numbers, or API keys before the data is indexed and made available for retrieval.[9]

Layer 2: Data discovery and classification

Native discovery: Using Elasticsearch's powerful search and natural language processing (NLP) capabilities, security teams can run queries and analysis jobs to scan indexed data which includes application logs, retrieval augmented generation (RAG) documents, and LLM conversations for patterns matching PII, credentials, and other sensitive information. The open source community has developed resources, such as the elastic-pii project on GitHub, that provide models and pipelines for this purpose.[10]

Synergistic defense: For more advanced requirements, Elastic can be integrated with specialized data classification tools like Nightfall, BigID, or Informatica.[11] These tools can scan data within Elasticsearch, apply classification labels (e.g., “PII,” “Confidential”), and write this metadata back to the documents. This enriched data can then be used to drive automated security policies within Elastic, such as applying document-level security to all data tagged as sensitive.

Layer 3: Detection of downstream credential misuse

If sensitive credentials are leaked by an LLM, the attacker's next step is to use them. Elastic Security excels at detecting this subsequent malicious activity, providing a critical safety net.

Prebuilt detection rules: Elastic Security ships with a comprehensive library of over 1,000 prebuilt detection rules mapped to the MITRE ATT&CK® framework. This includes rules specifically designed to detect common credential access techniques, such as the use of tools like Mimikatz and Lazagne, or unauthorized access to sensitive credential stores like the Windows Credential Manager or Chrome's Login Data file.[12]

Correlated detections: Using Event Query Language (EQL), security teams can create high-fidelity rules that correlate different stages of an attack. For instance, a rule can detect remote access to a server (e.g., via SMB) followed by access to the Security Account Manager (SAM) registry hive, a clear indicator of an attempt to dump credentials.[12]

Layer 4: Secure architecture and anonymization

The architecture of the Elastic AI Assistant serves as a best-practice model for secure LLM integration. It features sophisticated and customizable field-level anonymization capabilities, ensuring that sensitive details within security alerts (which are used as context for the LLM) are obscured before being processed.[13] This principle of anonymizing context data is a crucial design pattern for mitigating disclosure risks in any LLM application.

LLM03:2025 Supply Chain

Vulnerability deep dive

The LLM supply chain presents a complex and expanded threat surface compared to traditional software. Vulnerabilities can arise not only from third-party code dependencies but also from the very components that constitute the AI model itself: pretrained models, training and fine-tuning datasets, and modular components like Low-Rank Adaptation (LoRA) adapters.[1] Key risks include the use of vulnerable open source packages, the ingestion of poisoned or backdoored pretrained models (as demonstrated by the PoisonGPT attack), reliance on models with weak or falsified provenance, the application of malicious LoRA adapters, and compromises within the CI/CD pipeline where models are built and deployed. Furthermore, unclear terms and conditions from model providers can create a risk of proprietary data being absorbed into model training without consent.[1]

The Elastic mitigation strategy: Visibility and integrity from code to cloud

A secure supply chain requires end-to-end visibility and integrity checks. Elastic's Search AI Platform serves as the central nervous system for this process, ingesting data from across the MLOps lifecycle to provide a unified view of risk.

Layer 1: Vulnerability and dependency management

Elastic provides a single pane of glass for viewing vulnerabilities across the entire application stack, from the code to the cloud infrastructure it runs on.

SCA and scanner integration: Elastic Security seamlessly integrates with leading Software Composition Analysis (SCA) and container scanning tools like **Snyk** and **Trivy**.[14] By ingesting vulnerability reports from these scanners into Elasticsearch, organizations can create a centralized, searchable inventory of all known vulnerabilities in their application code, dependencies, and container images.

Cloud native vulnerability management (CNVM): Elastic's native CNVM capability, which leverages the open source scanner Trivy, automatically scans cloud workloads and container images for vulnerabilities.[15] This is critical for securing the runtime environment of the LLM application and its components. This unified view allows teams to correlate vulnerabilities found in source code with those present in the production environment, enabling more effective prioritization.

Layer 2: Software and Model Bill of Materials (SBOM/MBOM) analysis

Rapid zero-day response: When a new vulnerability (like Log4Shell) is disclosed, security teams can instantly query their SBOM repository in Elasticsearch to identify every application and service that uses the vulnerable component. This capability dramatically reduces the time to identify exposure from days or weeks to mere seconds.

Dynamic and runtime analysis: The industry is recognizing the limitations of static, point-in-time SBOMs and is moving toward dynamic or runtime SBOMs that reflect what is actually executing in production.[16] The Search AI Platform is perfectly suited for this evolution. By correlating the data from a static SBOM with live process and library information collected by Elastic Defend or Elastic APM, security teams can differentiate between a vulnerability that is merely present in a container image and one that is active in a running process. This allows for far more intelligent risk prioritization, focusing remediation efforts on the most immediate threats.

Layer 3: Model and data integrity monitoring

To mitigate the risk of direct model tampering or the poisoning of local datasets, continuous integrity monitoring is essential.

File integrity monitoring (FIM): Elastic Agent's FIM capability can be configured to monitor critical AI/machine learning (ML) assets, including model files (e.g., .safetensors, .bin), configuration files, and directories containing training or RAG data.[17] Any unauthorized creation, modification, or deletion of these files will trigger an alert.

Correlated change auditing: While basic FIM on some operating systems may not capture user attribution, it can be powerfully combined with other data sources. By enabling host-level auditing with tools like Auditbeat or Sysmon (or using the eBPF backend on modern Linux kernels), file change events can be correlated with the specific user and process responsible, providing a complete audit trail for forensic investigation.[18] This directly counters threats like an attacker modifying a model file on disk to insert a backdoor.

Layer 4: CI/CD pipeline security

The MLOps pipeline is a high-value target for supply chain attacks.[19] Elastic provides the observability needed to secure this critical infrastructure.

Pipeline log ingestion: Ingest logs and events from CI/CD platforms like GitHub and GitLab into Elastic using available integrations and community-built tools.[20]

Anomaly detection in MLOps: Use Elastic's anomaly detection to monitor for suspicious activity within the pipeline.

LLM04:2025 Data and Model Poisoning

Vulnerability deep dive

Data and Model Poisoning involves the manipulation of data used during pretraining and fine-tuning or for RAG systems to introduce vulnerabilities, biases, or hidden backdoors.[1] This attack differs from general supply chain issues by focusing specifically on corrupting the *data* that shapes the model's behavior.

The Elastic mitigation strategy: Detecting the symptoms and effects

The Elastic strategy focuses on two key areas: detecting the *consequences* and *symptoms* of a poisoned model in production, and robustly protecting the data assets that an organization does control, such as the knowledge base in a RAG architecture.

A poisoned model will, by definition, exhibit anomalous behavior when triggered. It may produce biased or toxic outputs, provide factually incorrect information, or consume resources in an unusual way. These are all observable deviations from a normal baseline. By leveraging Elastic's powerful machine learning capabilities to monitor model outputs and performance metrics, security teams can detect the symptoms of poisoning even if the poison itself is invisible.

Layer 1: Monitoring for behavioral anomalies in production

Comprehensive output logging: Ingest all LLM prompts and responses into Elastic. This data should be enriched with additional metadata, such as sentiment scores, toxicity classifications (from integrated third-party tools or custom models), and response length.[3]

ML-powered anomaly detection: Create Elastic ML anomaly detection jobs to continuously monitor these output characteristics over time. A sudden, statistically significant shift — such as a spike in toxic responses, a deviation in average sentiment for a specific topic, or a model abruptly refusing to answer certain types of questions — can be a strong indicator of a poisoning attack or a triggered backdoor.[6]

Performance anomaly detection: Monitor key performance indicators like latency and token usage. A poisoned model, when its trigger is activated, might enter a computationally expensive loop or generate an excessively long response. This would manifest as a detectable spike in the gen_ai.performance.duration or gen_ai.usage.completion_tokens metrics, which can be flagged by an ML job.[21]

Layer 2: Protecting the RAG knowledge base

In a RAG architecture, the vector database and its underlying source documents are a prime target for data poisoning.[1]

Integrity controls: Use **file integrity monitoring (FIM)** to continuously watch the directories where source documents and vector database files are stored. An alert on an unauthorized file modification can prevent poisoned data from being ingested and indexed into the RAG system.[17]

Access controls: Implement strict access controls on the Elasticsearch indices that constitute the RAG knowledge base. Use Elasticsearch's native **role-based access control (RBAC)** to create roles that grant write permissions only to specific, authorized data ingestion processes.

Layer 3: Evaluating RAG relevance and groundedness

A poisoned document in a RAG system will likely result in a generated answer that is poorly grounded in the retrieved context or factually incorrect. Monitoring the quality of the RAG process itself can thus serve as a proxy for detecting data poisoning.

Synergistic defense with evaluation frameworks: Integrate the RAG application with evaluation frameworks like **DeepEval** or **RAGAs**.[22] These tools can programmatically assess metrics like contextual precision, contextual recall, and answer relevancy.

Monitoring quality scores: Log these quantitative evaluation scores into Elasticsearch. By creating ML anomaly detection jobs or simple threshold-based alerts on these metrics, the MLOps team can be automatically notified of a sudden degradation in RAG quality.

LLM05:2025 Improper Output Handling

Vulnerability deep dive

Improper Output Handling occurs when an application fails to properly validate, sanitize, and handle the content generated by an LLM before passing it to downstream components or systems. This vulnerability is particularly dangerous because LLM outputs can be controlled by an attacker via prompt injection (LLM01). If the application implicitly trusts the LLM's output, it creates a "confused deputy" scenario, where the LLM application is tricked into attacking other parts of the infrastructure. Successful exploitation can lead to a wide range of classic web application vulnerabilities, including Cross-Site Scripting (XSS), SQL Injection (SQLi), Server-Side Request Forgery (SSRF), and even Remote Code Execution (RCE) on backend systems.[1]

The Elastic mitigation strategy: Tracing and detecting downstream exploitation

The core of this vulnerability lies in the broken trust boundary between the LLM application and its downstream dependencies. Elastic's primary strength in mitigating this risk is its ability to provide complete, end-to-end visibility across these boundaries, tracing the flow of tainted data from the LLM to the point of impact and detecting the resulting attack.

The mitigation strategy follows a clear logical progression. An attacker first uses prompt injection to make the LLM generate a malicious payload. The vulnerable application then fails to sanitize this output and embeds it in a query to a backend database. Elastic APM, with distributed tracing enabled, captures this entire transaction flow. It creates a single, correlated trace that begins at the LLM application, shows the database call with the malicious payload visible in the span's metadata, and records the subsequent database error or anomalous response time. Simultaneously, Elastic Security's prebuilt detection rules, running on data from the database server, can identify and alert on the SQL injection attempt itself. By correlating the APM trace with the security alert, an analyst receives a complete, unambiguous picture of the attack: what happened (SQLi), how it happened (via the LLM's unsanitized output), and why it happened (the initial prompt that triggered it).

Layer 1: End-to-end visibility with distributed tracing

Full-stack instrumentation: The first step is to instrument the LLM application and all its downstream services (backend APIs, databases, message queues) with Elastic APM agents or compatible OpenTelemetry agents.[23]

Distributed tracing: Distributed tracing automatically stitches together the individual operations of a request as it flows through a microservices architecture into a single, unified view.[24]

Root cause analysis: When an attack based on improper output handling occurs, the resulting APM trace provides an invaluable forensic artifact. Analysts can visually follow the transaction from the LLM to the exploited downstream service. By inspecting the metadata of each span in the trace, they can see the exact data being passed between components, drastically reducing mean time to resolution (MTTR).

Layer 2: Detecting the exploit on the target system

APM provides visibility into the attack path, and Elastic Security provides detection at the point of impact.

Endpoint and cloud protection: Deploy **Elastic Defend** on all backend servers and cloud workloads. Elastic Defend provides deep visibility into host-level activity, including process execution, network connections, and file access.

Prebuilt detection rules: Leverage Elastic Security's extensive library of **prebuilt detection rules** to detect the final stage of the exploit.[25] This library contains thousands of rules mapped to MITRE ATT&CK, including specific detections for:

- SQL Injection patterns in database logs
- Suspicious child processes spawned by web servers, indicating RCE
- Anomalous outbound network connections from application servers, indicating SSRF
- Path traversal attack patterns (../) in file access events

These rules provide immediate, high-fidelity alerts when the LLM's malicious output is executed by a downstream component.

Layer 3: Auditing and monitoring LLM outputs

Monitor the LLM's outputs for potentially malicious content.

Centralized logging: As with other vulnerabilities, ensure all LLM-generated responses are logged to Elasticsearch for auditing and analysis.[3]

Content-based alerting: Create custom detection rules or ML jobs to scan the gen_ai.completion field for suspicious content. This could include searching for JavaScript snippets (<script>), SQL keywords (SELECT, INSERT, DROP), shell commands (/bin/bash), or file paths. Alerting on the presence of such content, especially if the application is not designed to generate it, can provide an early warning of a potential attack before the downstream system is exploited.

LLM06:2025 Excessive Agency

Vulnerability deep dive

Excessive Agency arises when an LLM-based system, often referred to as an "agent," is granted an inappropriate level of authority or autonomy to interact with other systems through tools, plugins, or APIs. The vulnerability is in the level of permissioning provided to the agent. The root causes fall into three categories: **Excessive Functionality**, **Excessive Permissions**, and **Excessive Autonomy**.[1] When an agent is compromised, is manipulated via prompt injection, or simply hallucinates, this excessive agency allows it to perform damaging actions like deleting data, executing financial transactions, or exfiltrating sensitive information.

The Elastic mitigation strategy: Monitoring the agent's every move

Securing LLM agents requires adopting Zero Trust Architecture (ZTA): The agent must be treated as a potentially untrustworthy entity whose every action is monitored and verified. Elastic provides the unified security and observability platform necessary to enforce ZTA, as well as the ability to capture a complete record of agent activity and use analytics to distinguish between legitimate and malicious behavior.

The core of this strategy is based on the understanding that an agent's actions are ultimately observable events — API calls, database queries, process executions, network connections. Elastic's comprehensive data collection capabilities can ingest this telemetry from across the environment. Normal agent behavior typically forms a predictable pattern. Any deviation from this baseline would be a high-severity anomaly.

Layer 1: Comprehensive agent activity monitoring

API and tool call monitoring: The first step is to capture a log of every action the agent takes. Elastic's integrations for cloud providers can capture service API call logs (e.g., AWS CloudTrail, Azure Monitor), while custom tools should be instrumented to produce structured logs that are shipped to Elastic.[3]

LLM observability for agentic workflows: Utilize Elastic's LLM observability capabilities to trace the entire agentic workflow.[26] This provides a detailed, correlated view of the sequence of user prompts, internal model reasoning steps (if available), and the resulting tool invocations.

Downstream system monitoring: Deploy Elastic Defend on the hosts where agents run to monitor their process and network activity.[27] Use Elastic APM to trace the agent's interactions with other internal microservices.[28]

Layer 2: Behavioral baselining and anomaly detection

With a complete record of agent activity, Elastic's machine learning can be used to detect abuse.

ML-powered anomaly detection: Create anomaly detection jobs to normalize LLM activity. These jobs can be based on several dimensions:

- **Anomalous tool use**: Detect when an agent calls a new or statistically rare API or uses a common API with unusual parameters.
- **Anomalous user context**: Correlate agent actions with the user who initiated the request.
- **Anomalous resource access**: An agent that normally only reads from a specific database table suddenly performing a DELETE or UPDATE operation is a critical anomaly that ML can detect.

Entity-centric risk analytics: Elastic's Advanced Entity Analytics can build dynamic risk profiles for users and hosts. Anomalous behavior detected from LLM agents can be fed into this engine, contributing to an overall risk score. A user whose requests consistently lead to anomalous agent behavior will see their risk score increase, flagging them for investigation.[6]

Layer 3: Detecting lateral movement and impact

A common goal for an attacker who has subverted an agent is to use it for **lateral movement** — pivoting from the agent's host to other systems in the network.

Lateral movement detection package: Elastic Security includes a dedicated package for detecting lateral movement. It uses machine learning models to identify suspicious file transfers over protocols like SMB and anomalous Remote Desktop Protocol (RDP) or SSH sessions, which are common attacker techniques.[29]

Prebuilt detection rules: Activate prebuilt rules that specifically target lateral movement TTPs, such as the abuse of Windows Management Instrumentation (WMI) for remote execution or dropping malicious payloads into remote startup folders.[30] If a compromised agent is used to stage a lateral movement attack, these dedicated detections will trigger, enabling rapid containment by the security team.

LLM07:2025 System Prompt Leakage

Vulnerability deep dive

System Prompt Leakage refers to the unauthorized disclosure of the initial instructions, configurations, and context provided to an LLM to guide its behavior. The core security risk is not the disclosure of the prompt's text itself, but rather the sensitive information or flawed security logic it might contain. OWASP rightly emphasizes that **the system prompt should not be considered a secret or used as a security control**. Leaked prompts become dangerous when they contain hardcoded secrets (API keys, passwords, connection strings) or describe security mechanisms (like permission models) that are improperly delegated to the non-deterministic LLM instead of being enforced by a separate, secure system.[1]

The Elastic mitigation strategy: Making the leak irrelevant

This strategy is based on a Zero Trust approach. A leaked system prompt is only harmful if it contains an exploitable secret, like an API key. According to security best practices, secrets should never be hardcoded but managed in an external vault. Second, if a key is leaked, its value to an attacker is nullified if its misuse can be immediately detected. Elastic Security can ingest logs from secret management systems and cloud provider key management services.[27] It includes prebuilt detection rules designed to spot anomalous secret access, such as a user or service retrieving a secret for the first time or from an unusual location.[31] This shifts the security focus from the initial, hard-to-prevent leak to the subsequent, high-confidence malicious action.

Layer 1: Detecting misuse of leaked information

If a system prompt containing a usable credential is leaked, an attacker's next step is to use it. This is the activity that Elastic is primed to detect.

Anomalous API and service usage: Ingest all API gateway logs and cloud provider audit logs (e.g., AWS CloudTrail, Azure Activity Logs) into the Elastic platform. Configure **Elastic ML anomaly detection jobs** to monitor this activity. An API key that is normally used only by production servers within a specific IP range suddenly being used from an unrelated geographic location is a strong indicator of compromise.[6]

Threshold-based alerting: Create threshold-based detection rules to alert on unusual volumes of API calls or access to highly sensitive endpoints.[32]

Layer 2: Auditing for sensitive data in prompts

While secure design is the primary mitigation, detective controls provide a valuable safety net and auditing capability.

Content scanning: As described in the mitigation for LLM02, Elastic's NLP capabilities and its ability to integrate with specialized data discovery tools can be used to continuously scan ingested LLM logs.[10] A custom detection rule can be written to search for sensitive data patterns (e.g., regex for API keys, password-like strings) within the gen_ai.model.instructions or gen_ai.prompt fields of the ECS schema.[3]

Developer feedback loop: When such a detection fires, it creates an immediate alert for the security and development teams.

Layer 3: Enforcing secure design through observability

The OWASP guidance states that critical security controls, such as privilege separation and authorization, must never be delegated to the LLM. Elastic Observability provides the tools to audit and enforce this principle.

Verifying authorization checks: By using Elastic APM with distributed tracing, architects and security reviewers can visualize the entire transaction flow of an LLM-powered request.[28] This allows them to verify that authorization checks are being performed by a dedicated, deterministic microservice before the LLM is called or before a tool is executed.

LLM08:2025 Vector and Embedding Weaknesses

Vulnerability deep dive

This category of vulnerabilities is centered on the unique security challenges of RAG systems, which are becoming the standard for grounding LLMs in factual, proprietary data.[1] The vector database and the embeddings it stores can introduce new attack vectors:

Unauthorized access and data leakage: The most significant risk is a failure to enforce access controls within the vector database. In a multitenant environment, this can allow one user's query to retrieve and expose sensitive documents belonging to another user.

Data poisoning: An attacker can inject malicious or manipulated documents into the RAG knowledge base. When retrieved, this poisoned context can be used to trick the LLM into generating misinformation or performing harmful actions.

Embedding inversion: Adversary analyzes retrieved embeddings to reverse-engineer and reconstruct the original sensitive source data from which they were generated, compromising confidentiality.

The Elastic mitigation strategy: Secure and observable RAG with Elasticsearch

The fundamental security challenge in RAG is ensuring that a user's query only retrieves context from documents they are explicitly authorized to view. This requires a mechanism to filter search results based on user identity or role at query time. Elasticsearch provides this capability natively through **document-level security (DLS) and field-level security (FLS)**.[33]

Layer 1: Enforcing permission-aware RAG with native security

Elasticsearch as the vector database: The first step is to leverage Elasticsearch as the vector database, which offers powerful hybrid search capabilities, combining traditional full-text search with dense vector search for optimal relevance.

Role-based access control (RBAC): Use Elasticsearch's built-in security to define user roles with minimal required levels of access.

Document-level security (DLS): This is the critical component for secure RAG. During the data ingestion process, enrich each document with metadata fields that define its access permissions. Then, configure each user role with a DLS query that filters documents based on this metadata.

Preventing data leakage: When the RAG application queries Elasticsearch to retrieve context, this DLS filter is automatically applied. This guarantees that the context passed to the LLM only contains information the end user is authorized to see.[33]

Layer 2: Monitoring for data poisoning and inversion attacks

Data poisoning mitigation: Use file integrity monitoring to protect the source documents and vector database files on disk, and use RBAC to strictly control write access to the RAG index via the API.[17] Additionally, monitor RAG quality and relevance metrics for sudden degradation, which could indicate a poisoning attack.[22]

Embedding inversion detection: The mitigation strategy focuses on identifying the precursor activity: anomalous query patterns. An attacker attempting to invert embeddings will likely need to issue a large volume of very specific, slightly varied k-Nearest Neighbor (kNN) vector search queries.

- **Audit and query logging**: Enable and ingest Elasticsearch audit logs and slow query logs into a separate monitoring index.
- **ML-powered anomaly detection**: Create an Elastic ML anomaly detection job to monitor this query activity. The job can learn the baseline query behavior for each user or service. A sudden, high-volume burst of kNN queries from a single source, deviating significantly from its normal pattern, would be flagged as a high-severity anomaly, providing a strong signal of a potential inversion attack in progress.[6]

Layer 3: RAG performance and observability

Beyond security, the performance of the RAG pipeline is critical for user experience.

End-to-end observability: Use Elastic Observability to monitor the performance of the entire RAG application, from the user's browser to the final LLM response.[34]

Performance metrics: Track key metrics such as retrieval latency, generation latency, and token usage. This allows teams to identify performance bottlenecks, whether they lie in the retrieval or generation step, and to optimize costs.[35]

Native integration: Because RAG is a first-class workload in Elastic, the platform provides deep insights into the performance of the retrieval step, including the performance of vector search, keyword search, and hybrid search configurations, allowing for fine-grained tuning and optimization.[36]

LLM09:2025 Misinformation and Overreliance

Vulnerability deep dive

Misinformation occurs when an LLM generates "hallucinations" — outputs that are plausible-sounding but are factually incorrect, fabricated, or nonsensical. This core unreliability is compounded by the human tendency for **over-reliance**, where users place undue trust in the LLM's output without performing necessary verification. This combination can have severe consequences, as demonstrated by real-world incidents where an airline's chatbot provided incorrect policy information leading to a successful lawsuit, or a lawyer submitted a legal brief citing entirely fake cases generated by an LLM. The risk also extends to technical domains, such as unsafe code generation, where an LLM might confidently suggest using insecure or even nonexistent software libraries, leading to security vulnerabilities if trusted by a developer.[1]

The Elastic mitigation strategy

The most effective technical mitigation for LLM hallucination is **RAG**, which grounds the model's responses in a corpus of factual, verifiable data. Elastic, with its best-in-class vector and hybrid search capabilities, is a premier platform for building these RAG applications.[33]

However, simply implementing RAG is not enough; its effectiveness must be continuously monitored. Key metrics like contextual recall and precision are vital for understanding if the RAG system is retrieving relevant information.[22] These metrics can be calculated by external evaluation frameworks and logged into Elastic. From there, Elastic's ML and alerting capabilities can monitor these quality scores over time, creating a closed-loop system that alerts developers when the RAG system's performance degrades, which is a leading indicator of increased misinformation risk.

Layer 1: Powering reliable RAG systems

Advanced retrieval with Elasticsearch: As detailed in the mitigation for LLM08, using **Elasticsearch as the vector database** for a RAG implementation provides sophisticated **hybrid search**, which often yields more relevant and accurate context for the LLM than vector search alone. Better context leads to better, more factual answers and fewer hallucinations.[33]

Trusted knowledge bases: Use Elastic to index and search curated, up-to-date, and trusted internal knowledge sources. This ensures that the LLM is grounded in information that has been vetted by the organization.

Layer 2: Monitoring RAG and output quality

Synergistic defense with evaluation frameworks: To quantitatively measure the effectiveness of the RAG pipeline, integrate the application with evaluation frameworks like **DeepEval** or **RAGAs**.[22] These tools can programmatically assess the quality of the retrieval and generation process.

Logging quality metrics: Continuously log key quality metrics into Elasticsearch. These should include:

- **Contextual precision**: Are the most relevant documents ranked highest in the retrieved context?
- **Contextual recall**: Did the retrieval process find all the necessary documents to answer the question?
- **Answer relevancy**: Is the final generated answer relevant to the user's original query?

Quality monitoring and alerting: Create Kibana dashboards to visualize these quality metrics over time. This provides an ongoing view of the RAG system's health. More importantly, configure **Elastic ML anomaly detection jobs or threshold-based alerts** on these metrics. An automated alert triggered by a sudden drop in contextual recall or answer relevancy can notify the MLOps team of a performance degradation, allowing them to investigate and remediate the issue before it leads to widespread misinformation.

Layer 3: Enabling human-in-the-loop and feedback

Technical monitoring alone is not sufficient; human oversight is a critical component of managing misinformation risk.

Capturing user feedback: Implement feedback mechanisms within the LLM application, such as "thumbs up/thumbs down" buttons or a way for users to report inaccurate answers.

Actionable insights from feedback: Use Kibana's powerful data visualization and analysis tools to explore this feedback data. This insight is crucial for prioritizing which documents in the RAG knowledge base need to be updated or where fine-tuning efforts should be focused to improve the model's accuracy.

LLM10:2025 Unbounded Consumption

Vulnerability deep dive

Unbounded Consumption is a critical vulnerability that occurs when an application allows for excessive and uncontrolled use of an LLM's computational resources. Attackers can exploit this vulnerability to cause not only service degradation or outages (DoS) but also financial losses for the organization, a tactic known as **Denial of Wallet (DoW)**.[37] This attack specifically targets the pay-per-token or pay-per-inference billing models common to cloud-hosted LLM services. Common attack vectors include submitting a high volume of requests, feeding the model exceptionally long or complex inputs that require more processing, or crafting queries designed to trigger resource-intensive operations.[1]

The Elastic mitigation strategy: Comprehensive monitoring, anomaly detection, and alerting

This is a core competency of Elastic Observability. The attack itself manifests as a spike in specific, measurable metrics: API call frequency, token counts, CPU/memory utilization, and, ultimately, cost.[38] Elastic's dedicated integrations for major LLM providers like OpenAI, AWS Bedrock, and Google Vertex AI are designed to collect these exact metrics out-of-the-box.[26] Elastic's machine learning capabilities can then analyze these time-series metrics and flag any statistically significant deviations.[6] Finally, the Kibana Alerting framework can be configured to take action when an ML anomaly is detected or when a key metric crosses a predefined budget threshold.[39] This creates an end-to-end solution: Ingest the critical metrics, use ML to intelligently detect an attack, and alert the responsible team to take immediate action, such as blocking the attacking IP address or enabling stricter rate limiting.

Layer 1: Real-time usage and cost monitoring

Out-of-the-box integrations: Deploy Elastic's integrations for **OpenAI**, **AWS Bedrock**, **Azure OpenAI**, and **GCP Vertex AI**. These integrations come with prebuilt Kibana dashboards that provide immediate, comprehensive visibility into key consumption metrics without requiring custom configuration.[34]

Key metrics tracking: Continuously monitor crucial metrics that directly relate to consumption and cost:

- gen_ai.usage.prompt_tokens
- gen_ai.usage.completion_tokens
- gen_ai.usage.total_tokens

These metrics provide a real-time view of token consumption, which is often directly tied to cost.[21]

Request and error rate monitoring: Track the number of API invocations and the rate of errors. A classic DoS or DoW attack will often appear as a massive, sudden spike in the request rate, which can be easily visualized and monitored on these dashboards.[40]

Layer 2: Anomaly detection for resource spikes

While static thresholds are useful, they can be noisy. Machine learning provides a more intelligent layer of detection that understands the normal context and seasonality of your application's workload.

ML-powered anomaly detection: Create **Elastic ML anomaly detection jobs** on key consumption and performance metrics. This is more powerful than simple thresholding because it learns what is "normal" for your application at different times of the day or week, reducing false positives.[6]

Targeted anomaly jobs: Set up specific jobs to detect anomalies in:

- **Token usage**: A sudden, sustained increase in the number of tokens per request, per user, or per source IP address.
- **API call frequency**: A volumetric attack will manifest as a statistically significant spike in the rate of API calls.
- **Infrastructure metrics**: For self-hosted LLMs, use Metricbeat to monitor host-level metrics. A resource-intensive query designed to exhaust the system will cause a detectable spike in CPU and memory utilization on the inference servers.[41]

Layer 3: Proactive "Denial of Wallet" alerting

The most direct mitigation for the financial risk of DoW attacks is to create explicit financial guardrails using the alerting framework.

Kibana Alerting framework: Use the powerful and flexible Kibana Alerting framework to create proactive alerts based on consumption metrics.[39]

Budget-based threshold rules: Create a threshold-based rule that directly monitors the financial risk. For example, configure a rule to calculate the cumulative sum of gen_ai.usage.total_tokens over a specific time window (e.g., 24 hours). Set the threshold for this rule to a value that corresponds to your daily budget for the LLM service.

High-priority notifications: Configure the alert to trigger a high-priority action, such as sending a notification to a PagerDuty or Slack channel monitored by the operations and finance teams. This ensures that when a potential DoW attack is causing costs to approach the daily budget, the right people are notified immediately and can take action — like blocking the source of the traffic or temporarily disabling the service — before the budget is exceeded and financial damage occurs. This provides a direct, business-relevant mitigation for the Denial of Wallet attack vector.

Conclusion

The advent of generative AI has introduced a new and formidable set of security challenges, as codified by the OWASP Top 10 for LLM Applications. Addressing these risks requires a fundamental shift away from siloed, single-purpose security tools toward a unified platform that can provide comprehensive visibility and correlated insights across the entire AI application ecosystem.

As this report has demonstrated, the Search AI Platform, with its unique fusion of security analytics and deep observability, provides the necessary foundation to meet this challenge head-on. By leveraging Elastic, organizations can:

Achieve end-to-end visibility: From user prompts and LLM responses to API calls, database queries, and underlying infrastructure performance, Elastic ingests and normalizes the full spectrum of telemetry required for robust AI security.

Move beyond static signatures: Employ powerful machine learning and behavioral analytics to detect the subtle and dynamic attacks characteristic of the LLM threat landscape, such as prompt injection, model poisoning, and excessive agency.

Correlate cause and effect: Use the power of EQL and distributed tracing to connect suspicious LLM activity with its impact on downstream systems, transforming low-fidelity signals into confirmed, actionable security alerts.

Secure the RAG pipeline: Leverage native, enterprise-grade security features like document-level security within the Elasticsearch vector database to build inherently secure, permission-aware RAG applications.

Implement proactive guardrails: Utilize real-time monitoring and a flexible alerting framework to establish proactive guardrails against operational and financial risks like Denial of Wallet attacks.

Ultimately, confidently building, deploying, and scaling the next generation of AI-powered applications requires a platform that is as integrated, intelligent, and adaptable as the AI systems it is designed to protect. Elastic's Search AI Platform provides this comprehensive, defense-in-depth foundation, enabling organizations to innovate boldly while effectively managing the full spectrum of risks outlined in the OWASP Top 10 for LLMs.

 |

elastic

Works Cited

1. OWSAP, “OWASP Top 10 for LLM Applications 2025,” https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025, 2024.
2. NeuralTrust, “How to Set Up Prompt Injection Detection for Your LLM Stack,” https://neuraltrust.ai/blog/prompt-injection-detection-llm-stack, 2025.
3. Elastic Security Labs, “Elastic Advances LLM Security with Standardized Fields and Integrations,” https://www.elastic.co/security-labs/elastic-advances-llm-security, 2024.
4. Elastic Security Labs, “Embedding Security in LLM Workflows: Elastic's Proactive Approach,” https://www.elastic.co/security-labs/embedding-security-in-llm-workflows, 2024.
5. Elastic Docs, Inspect log anomalies, https://www.elastic.co/docs/solutions/observability/logs/inspect-log-anomalies.
6. Elastic Docs, Anomaly detection, https://www.elastic.co/docs/solutions/security/advanced-entity-analytics/anomaly-detection.
7. Elastic Security Labs, “Embedding Security in LLM Workflows: Elastic's Proactive Approach,” https://www.elastic.co/security-labs/embedding-security-in-llm-workflows, 2024.
8. Elastic Docs, Application data security, https://www.elastic.co/docs/solutions/observability/apm/secure-data.
9. DataSunrise, “Data Masking in Elasticsearch,” https://www.datasunrise.com/knowledge-center/data-masking-in-elasticsearch.
10. GitHub, https://github.com/bvader/elastic-pii.
11. Nightfall, “Guide to Cloud Data Loss Prevention (DLP) on Elasticsearch,” https://toolkit.nightfallai.com/guides/elasticsearch.
12. Elastic Security Labs, “Detect Credential Access with Elastic Security,” https://www.elastic.co/security-labs/detect-credential-access, 2023.
13. Elastic Blog, “Built-in protections for Elastic AI Assistant,” https://www.elastic.co/blog/combating-llm-threat-techniques-with-elastic-ai-assistant, 2024.
14. Elastic Docs, Snyk Integration, https://www.elastic.co/docs/reference/integrations/snyk.
15. Elastic Docs, Cloud native vulnerability management, https://www.elastic.co/docs/solutions/security/cloud/cloud-native-vulnerability-management.
16. Oligo, “5 SBOM Generation Tools & 5 Critical Best Practices,” https://www.oligo.security/academy/5-sbom-generation-tools-5-critical-best-practices, 2025.
17. Elastic Docs, File Integrity Module, https://www.elastic.co/docs/reference/beats/auditbeat/auditbeat-module-file_integrity.
18. Reddit, File Integrity Monitoring : r/elasticsearch, https://www.reddit.com/r/elasticsearch/comments/1k6gtrk/file_integrity_monitoring.
19. Elastic, What is MLOps?, https://www.elastic.co/what-is/mlops.
20. Airbyte, “How to Load Data from GitHub to ElasticSearch?,” https://airbyte.com/how-to-sync/github-to-elasticsearch-destination.
21. Elastic Observability Labs, “LLM observability: Track usage and manage costs with Elastic's OpenAI integration,” https://www.elastic.co/observability-labs/blog/llm-observability-openai, 2025.
22. Elasticsearch Labs, “How to optimize RAG retrieval in Elastisearch with DeepEval,” https://www.elastic.co/search-labs/blog/rag-retrieval-elasticsearch-deepeval, 2025.

23. Elastic Docs, Anonymous authentication, https://www.elastic.co/docs/solutions/observability/apm/anonymous-authentication.
24. GitHub, https://github.com/nipwaayoni/elastic-apm-php-agent/blob/master/docs/examples/distributed-tracing.md.
25. Elastic Docs, Prebuilt rule reference | Elastic Security Solution [8.18], https://www.elastic.co/guide/en/security/current/prebuilt-rules.html.
26. Elastic Docs, LLM observability, https://www.elastic.co/docs/solutions/observability/applications/llm-observability.
27. Elastic Blog, "What's new in Elastic Security 7.15: End threats at the endpoint...and beyond," https://www.elastic.co/blog/whats-new-elastic-security-7-15-0, 2021.
28. Atatus, "Understanding APM and Distributed Tracing in the Observability Stack," https://www.atatus.com/blog/apm-and-distributed-tracing, 2025.
29. Elastic Docs, Lateral Movement Detection Model, https://www.elastic.co/docs/reference/integrations/lmd.
30. Elastic Docs, Lateral Movement via Startup Folder | Prebuilt detection rules reference, https://www.elastic.co/docs/reference/security/prebuilt-rules/rules/windows/lateral_movement_via_startup_folder_rdp_smb.
31. GitHub, https://github.com/elastic/detection-rules/blob/main/rules/integrations/aws/credential_access_new_terms_secretsmanager_getsecretvalue.toml.
32. Saidani Mohamed El Amine, "Elastic Detection Rule 101," https://mohamedsaidani34.medium.com/elastic-detection-rule-101-d4ecfac6700f, 2023.
33. Elastic Docs, RAG, https://www.elastic.co/docs/solutions/search/rag.
34. Elastic, LLM observability, https://www.elastic.co/observability/llm-monitoring.
35. DZone, "Enhanced Monitoring Pipeline With Advanced RAG Optimizations," https://dzone.com/articles/enhanced-monitoring-pipeline-rag-optimizations, 2025.
36. Elastic, "Elastic's RAG Based AI Assistant for Observability," https://www.youtube.com/watch?v=CDQtAg8YHsA, 2024.
37. Kalana Wijethunga, "Keep an Eye on Your Cloud Bill: How Denial of Wallet Attacks Affect Cloud Apps," https://kalanawijethunga.medium.com/keep-an-eye-on-your-cloud-bill-how-denial-of-wallet-attacks-affect-cloud-apps-42e94ba2d643, 2025.
38. Promptfoo, "Beyond DoS: How Unbounded Consumption is Reshaping LLM Security," https://www.promptfoo.dev/blog/unbounded-consumption, 2024.
39. Elastic, Kibana Alerting, https://www.elastic.co/kibana/alerting.
40. Elastic Observability Labs, "LLM Observability for Google Cloud's Vertex AI platform — Understand performance, cost and reliability," https://www.elastic.co/observability-labs/blog/elevate-llm-observability-with-gcp-vertex-ai-integration, 2025.
41. Elastic Docs, Detect metric anomalies, https://www.elastic.co/docs/solutions/observability/infra-and-hosts/detect-metric-anomalies.