opentext™

Protect AI-powered apps from emerging vulnerabilities

Get comprehensive security that identifies AI-related risks and secures applications across the SDLC

Benefits

- Security for AI-powered applications, not just traditional code
- Detection of AI- and LLM-specific vulnerabilities
- AI-aware testing across the full development lifecycle
- Practical risk reduction at DevSecOps speed

AI is powering a new generation of intelligent applications—but it's also introducing new vulnerabilities that traditional AppSec tools can't detect. As enterprises embed LLMs, AI agents, and model-driven features into customer-facing and mission-critical systems, they expose new attack surfaces, from prompt interfaces to AI output handling.

OpenText™ Application Security helps security leaders protect AI-powered applications across the entire SDLC. By extending proven SAST, SCA, and DAST capabilities with AI-specific detection, automated triage, and intelligent risk prioritization, OpenText helps teams reduce emerging threats, improve AppSec efficiency, and secure innovation at enterprise scale.

Security for AI-powered applications, not just traditional code

OpenText™ Application Security helps organizations secure the complete AI-powered application stack. This includes the code, APIs, data flows, and AI components that drive intelligent behavior. As large language models, AI agents, and GenAI services are integrated into customer-facing and mission-critical systems, we expand proven AppSec practices to support these modern architectures. The result is strong protection for AI functionality without slowing innovation.

Detection of AI- and LLM-specific vulnerabilities

Our AppSec portfolio addresses the unique security risks introduced by AI-powered features—threats that traditional testing methods often miss. These include prompt injection, insecure model interactions, data leakage through AI responses, unsafe output handling, and vulnerabilities in agent orchestration or tool invocation. We also identify conventional application and API flaws that can expose or worsen AI-related risks.

AI-aware testing across the full development lifecycle

OpenText supports continuous, AI-aware security testing from early development through CI/CD to production. By combining SAST, DAST, and software supply chain analysis with AI-assisted auditing and prioritization, OpenText gives teams clear visibility into how AI features interact with application logic, sensitive data, and external services. This helps catch and address risks before they reach users.

Practical risk reduction at DevSecOps speed

AI-powered triage and remediation streamline workflows by focusing teams on the vulnerabilities that matter most. By reducing false positives and highlighting real, exploitable risks—including those specific to AI—OpenText AppSec enables faster decision-making, stronger collaboration between security and development, and secure delivery of AI-driven software at scale.

Real-world AI vulnerabilities in action

AI features create new risks that traditional AppSec often misses—from prompt injection to unsafe outputs and vector-based data leaks. This table highlights real-world vulnerabilities in LLM-powered apps, mapped to the OWASP Top 10 for LLMs, showing how AI expands the attack surface and why AI-aware protection is critical across the SDLC.

Category	Description	OWASP Top 10 for LLM Apps
Anthropic misconfiguration: Unspecified token limit	Omitting token or rate limits for Anthropic APIs allows users to issue long or unlimited requests, leading to resource exhaustion and inflated costs.	LLM05:2025 Improper Output Handling/LLM10:2025 Unbounded Consumption
Credential management: Hardcoded API credentials	Embedding API keys or passwords directly in code exposes them to anyone who can view the source, enabling unauthorized access.	-
Cross-site scripting: AI	Reflecting AI-generated HTML or script output without proper sanitization enables attackers to run JavaScript in users' browsers.	LLM03:2025 Supply Chain/ LLM05:2025 Improper Output Handling
Cross-site scripting: DOM AI	Inserting untrusted AI output into the Document Object Model (DOM) on the client side lets malicious payloads execute by altering the DOM.	LLM03:2025 Supply Chain/ LLM05:2025 Improper Output Handling
Data poisoning: AI embedding	Malicious or poisoned data inserted into vector stores can skew semantic search and leak sensitive information when access controls are weak.	LLM04:2025 Data and Model Poisoning/LLM08:2025 Vector and Embedding Weaknesses
Data positioning: AI training	Attackers can corrupt training or fine-tuning datasets with biased or backdoored examples, so the model produces harmful outputs.	LLM04:2025 Data and Model Poisoning
Dynamic code evaluation: Unsafe TensorFlow deserialization	Deserializing Keras models using unsafe YAML loading (yaml. unsafe_load) enabled arbitrary code execution and was removed to mitigate the risk.	LLM03:2025 Supply Chain
Embedding and data exposure	Poorly partitioned embeddings allow other queries or tenants to retrieve or infer sensitive data stored in a vector database.	LLM08:2025 Vector and Embedding Weaknesses
Encoding confusion: Invisible characters	Hidden unicode characters in configuration files can mask malicious instructions.	LLM03:2025 Supply Chain

Category	Description	OWASP Top 10 for LLM Apps
Excessive agency	Granting LLM-powered agents broad permissions to call functions or external services increases the chance of misinterpretation or malicious behavior.	LLM06:2025 Excessive Agency
OpenAI misconfiguration: Unspecified token limit	Failing to bound response size or query rate for OpenAI calls leaves the system exposed to unbounded consumption and unexpected costs.	LLM05:2025 Improper Output Handling/LLM10:2025 Unbounded Consumption
Path manipulation	Allowing unchecked file path input lets attackers traverse directories (e.g., via "../") to access sensitive files.	LLM02:2025 Sensitive Information Disclosure/LLM05:2025 Improper Output Handling
Privacy violation	Unsanitized prompts or training data can expose personal or financial information, and prompt injection can coerce chatbots into retrieving secrets.	LLM02:2025 Sensitive Information Disclosure
Prompt injection	Carefully crafted input can override an LLM's instructions or extract data, making the model behave in unintended ways.	LLM01: 2025 Prompt Injection
Prompt injection: Persistent	Untrusted data stored in databases or logs that forms part of system prompts can persistently influence or compromise the model.	LLM01:2025 Prompt Injection
System information leak: Internal	Revealing file paths, usernames, or stack traces in error messages gives attackers insights into the environment.	LLM02:2025 Sensitive Information Disclosure
Unbounded consumption misconfiguration	Lack of rate limits or token bounds allows attackers to send large or numerous requests, draining resources and racking up costs.	LLM10:2025 Unbounded Consumption

Languages and frameworks supported

OpenText™ Application Security supports 33+ languages and frameworks, enabling consistent protection across modern, legacy, and AI-driven environments. From enterprise apps to AI workloads, this broad coverage helps teams detect both traditional vulnerabilities and emerging AI risks—wherever code is written or deployed.

This depth ensures AI-aware testing and proven AppSec controls apply across the entire development stack.

Language group	Examples
Enterprise and backend	Java, Kotlin, C#, .NET, VB.NET
Web and API development	JavaScript, TypeScript, Python, PHP, Ruby
Cloud and microservices	Go, Java, .NET, Node.js
AI/ML workloads	Python (TensorFlow, PyTorch), Java
Mobile applications	Android (Java/Kotlin), iOS (Swift, Objective-C)
Native and systems	C, C++
ERP and specialized	ABAP (SAP)
Infrastructure as code	Dockerfiles, Kubernetes YAML, Terraform

AI-driven development introduces new patterns of risk, but strong security principles still apply. OpenText™ Application Security extends trusted testing capabilities into AI-enabled architectures, combining broad language coverage, deep vulnerability detection, and AI-assisted analysis backed by continuously updated research.

By unifying testing across code, APIs, and open-source components, OpenText helps teams secure AI-powered applications with speed and consistency. Organizations can move faster with AI while keeping security tightly aligned to their development workflows and business priorities.

OpenText Application Security deployment options:

Accelerate cloud strategies with OpenText cloud experts

- OpenText Managed Private Cloud

Extend your team

- On-premises software, managed by your organization or OpenText

Run anywhere and scale globally in the OpenText public cloud

- OpenText Core Application Security (multi-tenant SaaS) consumed as a service [by pricing model such as subscription or by consumption]
- Example: OpenText Core Content runs in the OpenText Public Cloud with a user subscription

Run anywhere and scale globally in the hyperscaler cloud of your choice

- Hyperscaler cloud partners (OpenText Private Cloud, AWS private or public cloud, GCP private or public cloud, Azure private or public cloud)

Develop, connect, and extend your information management capabilities

- APIs from OpenText Developer Cloud

Resources

OpenText™ Application Security
Learn more ›

Watch a product demo
See it in action ›

Go deeper on AI and AppSec
Read blog ›

Understanding the quantum computing challenge
Read blog ›

Learn more about related capabilities
Learn more ›