Mend AI

Solution Architecture Brief

AI Security Lifecycle

1. Executive Summary

Mend AI is an AI security platform designed to address the unique risks AI introduces into modern applications, extending Mend.io's proven application security capabilities to this rapidly evolving attack surface.

As organizations adopt AI models, copilots, and agent-based architectures, security risks increasingly extend beyond traditional application vulnerabilities. AI systems introduce new attack surfaces through prompts, model interactions, agent behavior, data retrieval pipelines, tool access, and runtime decision-making, creating risks such as prompt injection, sensitive data exposure, insecure agent behavior, and runtime policy violations.

Mend AI delivers end-to-end coverage of the AI application lifecycle with a unified approach to discovering, assessing, and governing AI components, including models, AI frameworks, system prompts, and agent configuration files, across the organization's entire codebase. Support for AI agents, MCPs (Model Context Protocols), and agent tools is on the roadmap.

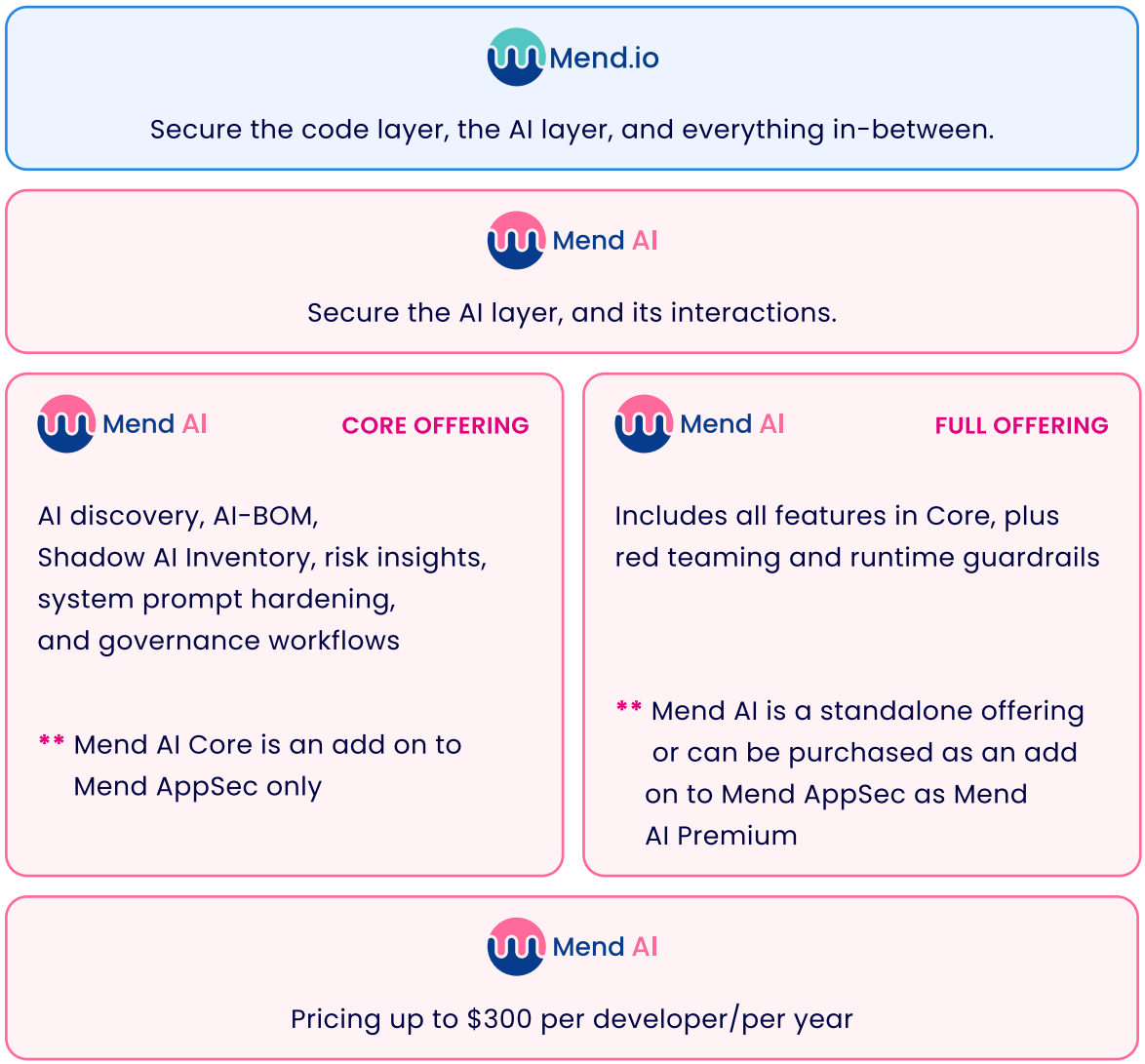
Unlike traditional application security tools that primarily focus on vulnerabilities in AI-generated code, Mend AI goes well beyond this coverage, securing the AI components within applications and the behavioral risks they introduce. While many standalone AI security tools operate primarily at either the model or runtime layer, Mend AI uniquely delivers protection across both the code and runtime layers to secure AI-powered applications end to end.

Mend AI operates through a shift-left philosophy, integrating AI security directly into existing key security touchpoints across development workflows via CI/CD pipelines, IDEs, and repository integrations to catch issues before they reach production.

Mend AI continues to expand with new capabilities, including Shadow AI Discovery, System Prompt Hardening, Red Teaming, AI Agent Configuration Scanning, AI Runtime Guardrails, and AI Policy and Governance. Grounded in Mend.io's experience securing large-scale software ecosystems, it is purpose-built to scale alongside both AI-driven development and enterprise growth.

2. Product Tiers and Packaging

Mend AI is available in two offerings, Core and Full, providing progressively deeper AI security capabilities. Mend AI Core is an add-on to Mend AppSec only. The Full offering includes all Core features plus red teaming and runtime guardrails, and can be purchased as a standalone offering or as an add-on to Mend AppSec (as Mend AI Premium).



3. Core Architecture

Mend AI is available in two offerings, Core and Full, providing progressively deeper AI security capabilities. Mend AI Core is an add-on to Mend AppSec only. The Full offering includes all Core features plus red teaming and runtime guardrails, and can be purchased as a standalone offering or as an add-on to Mend AppSec (as Mend AI Premium).

3.1 Detection Layer

The detection layer is the foundation of Mend AI. It automatically discovers all AI components embedded in application code through repository-level scanning.

Scanning Mechanism

Mend AI's discovery operates at the code level (not artifact level). Scanning is initiated via the Mend CLI or through Mend repository integrations (GitHub, Bitbucket, GitLab, Azure DevOps). The AI scan runs as a post-processing phase after the SCA scan completes, using a separate scanner that must be enabled for automatic AI discovery.

What Gets Detected

- **AI Models:** Third-party models sourced from Hugging Face and Kaggle, including closed-source model references from providers like OpenAI, Anthropic, and Google.
- **AI Frameworks:** Libraries like LangChain, LlamaIndex, TensorFlow, PyTorch, and other AI/ML frameworks are detected in dependency trees and code references.
- **System Prompts:** Foundational hidden instructions given to LLMs that define behavior, role, tone, and constraints. Automatically labeled as "conversational" for prioritization.
- **Agent Configuration Files:** Configuration files for AI agents are detected, providing visibility into how agents are defined and configured within the codebase.

On The Roadmap

- **AI Agents:** Autonomous or semi-autonomous AI components that use models and tools to perform tasks.
- **MCPs (Model Context Protocols):** Protocol implementations that connect AI models to external tools and data sources.

AI Bill of Materials (AI-BOM)

Every scan generates a continuously updated AI-BOM, a comprehensive inventory of all AI components in the codebase. The AI-BOM report includes the origin type (how the component was detected), the origin path (where it was found), associated risk factors, and model license risk as assessed by Mend's research team. Shadow AI components (previously unknown and unauthorized AI elements) are surfaced automatically.

3.2 Risk Assessment

Mend AI provides two categories of risk assessment: component-level risks that apply universally, and behavioral risks unique to each application.

AI Component Risk

System Prompt Risks: Identifies weaknesses in system prompts that could expose the application to prompt injection or other adversarial attacks. Each finding is scored using the proprietary AI Weakness Enumeration (AIWE) framework, which delivers a 1-100 risk score to help prioritize remediation.

System Prompt Hardening

System Prompt Hardening is the industry's first dedicated solution to detect, score, and automatically refine weaknesses in AI system prompts. Given that Gartner reports 32% of organizations experienced an attack using the application prompt in the past year, this capability addresses a critical attack vector.

The system operates through three mechanisms:

- **Detection and Visibility:** Identifies and surfaces embedded system prompts across all AI components, providing transparency into the directives shaping model behavior.
- **AI Weakness Enumeration (AIWE):** A proprietary scoring system modeled on the industry-approved CWSS framework, delivering a 1–100 score to quantify and prioritize AI security risks.

How it works

1. **Assign a unique identifier**
   ID: AIWE-<category>-<number>
2. **Determine category of risk**
   Identify the corresponding **OWASP Top 10 for LLM Applications** category.
3. **Calculate risk score**
   Using the **CWSS Scoring methodology** composed of three metrics Base Finding, Attack Surface, Environmental —each of which have an underlying score component. **Final score: Score = Base * ASurf * Env.**

BASE FINDING	ATTACK SURFACE	ENVIRONMENTAL
Internal Control Effectiveness	Required Privilege	Business Impact
Acquired Privilege	Required Privilege Layer	Likelihood Of Discovery
Acquired Privilege Layer	Access Vector	Likelihood Of Exploit
Internal Control Effectiveness	Authentication Strength	External Control Effectiveness
Finding Confidence	Level Of Interaction	Prevalence
	Deployment Score	
Base Finding Subscore	Attack Surface Subscore	Environmental Subscore

4. **Assign risk severity**
   critical: score ≥ 80 high: 60–79.9 medium: 40–59.9 low: < 40

WHY AIWE SCORING

- Consistent, standards-based method to quantify AI risks
- Empowers teams to prioritize and mitigate risks in AI system prompts
- Easily aligns various teams with clarity on the risk factors and steps to mitigate

Impact

- Streamline prioritization
- Accelerate MTTR
- Ensure security coverage
- Compliance readiness
- Improved communication
- and alignment

Agent Configuration Risks: Detects misconfigurations in the declarative files that define agent behavior before they reach production. Treated with the same rigor as application code, gives teams immediate visibility into the AI attack surface and enables CI-friendly enforcement directly within the Mend AI Scanner.

AI Dependency Risks: Scans AI models and frameworks to detect known CVEs, security advisories, and compromised or intentionally harmful AI components.

Hugging Face Unsafe Models: Surfaces confirmed unsafe Hugging Face models verified by Mend.io's vulnerability researchers.

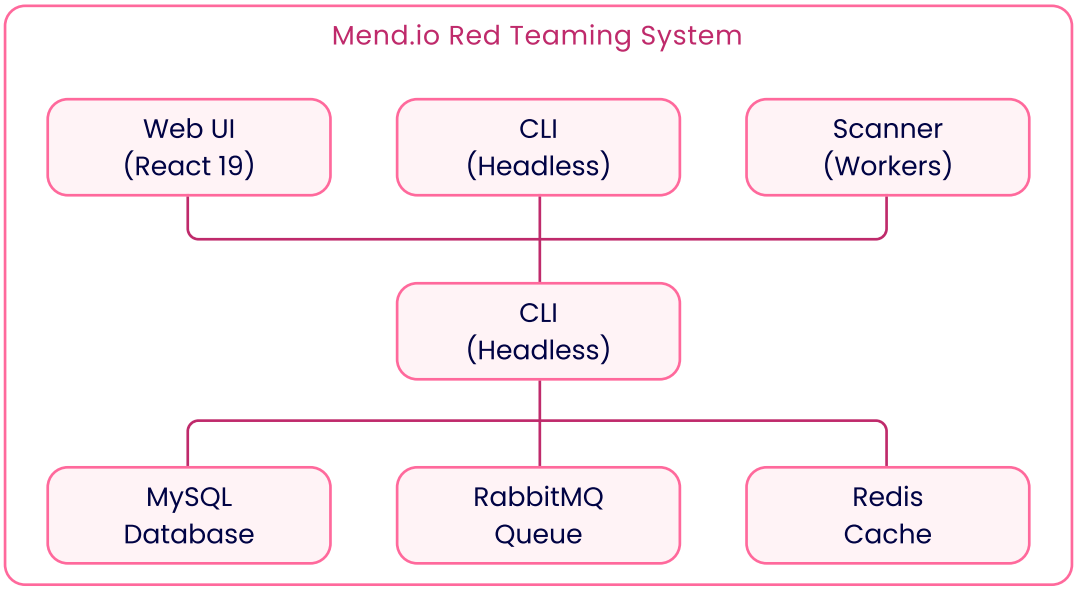
Licensing Risk: Evaluates AI model licenses for compliance obligations and restrictions on usage, redistribution, or modification, based on assessments by Mend.io's research team.

AI Behavioral Risks

AI Conversational Risks: Uncovers behavioral risks through red teaming, which simulates adversarial attacks against running AI-powered applications. Tests probe for prompt injection, context leakage, data exfiltration, jailbreak resistance, hallucinations, bias, harmful output generation, and model poisoning indicators.

Red Teaming

Red teaming provides dynamic security testing by simulating adversarial attacks against running AI-powered applications. Unlike static analysis, red teaming uncovers how AI systems actually behave under threat. Targets are configured per project, and tests are customizable to address the unique risks of each application.



Red teaming probes test for:

- **Sensitive data & PII exposure** — direct, session-based, social-engineering, and API/database vectors, plus cross-session data leakage.
- **System-prompt & internal exposure** — system-prompt extraction, internal tool discovery, and debug interface access.
- **Injection & access control** — SQL injection, command injection, and role-based access-control (RBAC) bypass.
- **Output integrity** — hallucination, unverifiable claims, overreliance, training-data leakage, and brand/identity imitation.
- **Harmful & off-policy content** — intellectual-property misuse, privacy violations, political bias, and teen-safety risks.
- **Custom intents & policies** — organization-defined adversarial goals and policy-compliance checks.
- **Attack strategies** - each probe is delivered through escalating techniques — direct prompting, prompt injection, jailbreak, gradual escalation (crescendo), and encoding/obfuscation (Base64, ROT13, homoglyphs, and more) — to surface weaknesses that simple prompts miss.

AI Runtime Risks: Enforces in-app runtime guardrails that screen prompts and responses at every LLM inference point, detecting policy violations, suspicious prompt behavior, anomalous agent activity, and unauthorized tool use in real time. Unlike proxy-based solutions, guardrails are embedded directly inside the application to protect against indirect prompt injection and secure both user-to-agent and agent-to-tool interactions.

3.3 Governance and Policy Workflow Engine

Mend AI leverages the same robust policy engine used by Mend AppSec, extended to address AI-specific concerns. Policies can be applied and enforced throughout the SDLC via automation workflows.

Policy enforcement areas include:

- **Model Usage Policies:** Control which AI models and providers are approved for use within the organization.
- **Licensing Compliance:** Enforce rules around AI model licenses, preventing use of models with incompatible terms.
- **Prompt Safety:** Set thresholds and requirements for system prompt security scores.
- **Risk Thresholds:** Define acceptable risk levels and automatically block or flag components that exceed them.
- **AI Agent Configuration Files:** Apply workflows that govern how AI agent configuration files are defined and deployed.

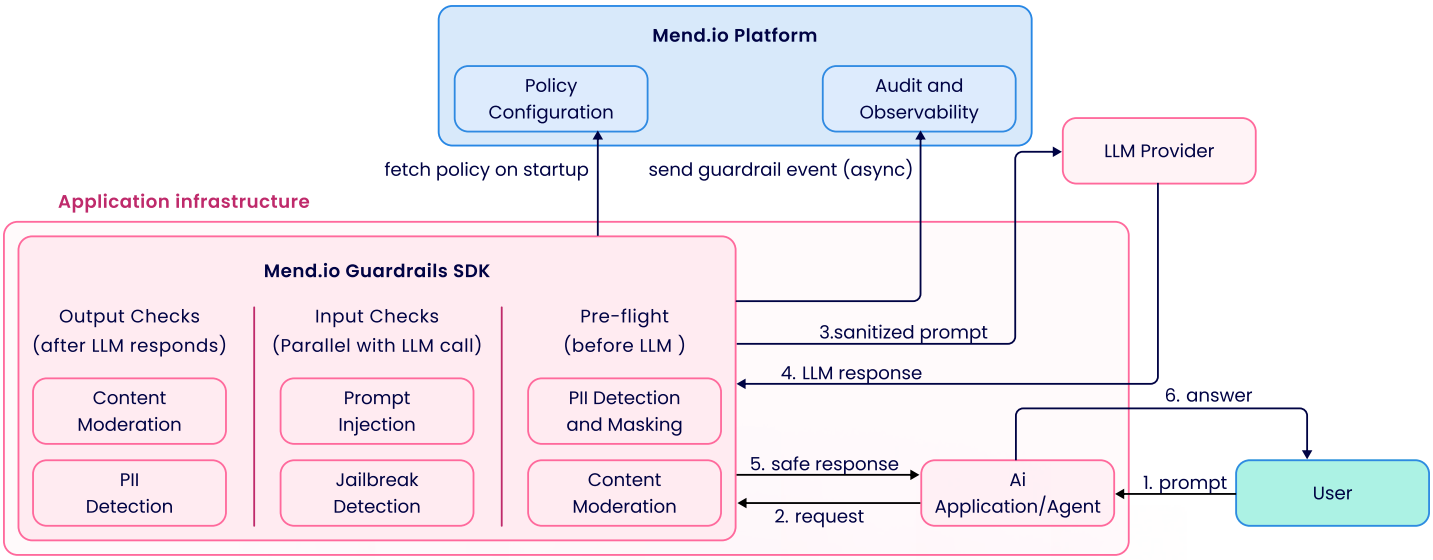
4. Runtime Protection Architecture

Mend AI's runtime guardrails is an in-app Python SDK that adds real-time AI safety enforcement directly within your application. Unlike proxy-based or SaaS guardrail solutions, all safety inference runs locally on your infrastructure, so prompt and response text never leave your application process.

4.1 Detection Layer

Every LLM call flows through up to three stages:

1. **Pre-flight:** Runs before the LLM call and can modify the message (e.g. mask PII) before it is sent.
2. **Input:** Runs in parallel with the LLM call, checking the user's prompt with zero added latency.
3. **Output:** Runs after the LLM responds, checking the generated text before it reaches the user.



Available Guardrails

- Prompt Injection: Detects injection attempts in user input before they reach the LLM.
- Jailbreak: Identifies attempts to bypass AI safety measures.
- Sensitive Data (PII): Detects and masks personal data in both inputs and outputs.

4.2 Integration Methods

Mend AI's guardrails integrate via a Python SDK using a drop-in client that wraps your existing LLM calls with minimal code changes. Developers replace their existing LLM client with the Mend Guardrails client, with no per-call-site changes required. The SDK is installed via pip and activated using a Mend API key configured in the platform.

For teams that prefer a server-based approach, the SDK also includes a Guardrails API Server that exposes an OpenAI-compatible endpoint, allowing integration without modifying application code directly.

Supported integrations include: Native SDK, OpenAI Agent SDK, OpenAI Compatible API, LangChain, Azure OpenAI, and LangFlow.

4.3 Runtime Capabilities

Runtime Dashboard and Events

The Mend Platform provides a dedicated AI Runtime view with a dashboard showing protected entities, total events by type (Alert, Block, Obfuscate), guardrail activity over time, and handled events by direction (inbound/outbound).

Event-Level Visibility

Every guardrail trigger is logged as a runtime event with details including: timestamp, protected entity, action taken, guardrail type, direction, detected activity, severity, and model name.

Policy Enforcement

Guardrail behavior is configured centrally per organization, with support for Alert, Block, and Obfuscate actions across four guardrail types: Prompt Injection, Jailbreak, PII, and Harmful Content.

Privacy and Data Control

No prompt or response text is ever sent to the Mend Platform. Only structured event metadata is reported (what happened, not what was said). For customers who need to correlate guardrail events with their own application data, each event carries a unique ID that can be accessed via callback hooks or OpenTelemetry integration. The customer has full and exclusive control over this data.

Resilience and Offline Support

The SDK continues to function locally if the Mend Platform is unreachable, and supports a fully offline mode for air-gapped or regulated environments.

5. Platform Integration Architecture

When purchased together, Mend AI is deeply integrated into Mend AppSec, sharing a unified UI and policy engine across SCA, SAST, Container, and AI security products

5.1 Integration Methods

Integration	Details
Mend CLI	Primary scanning interface. AI scan runs automatically after SCA scan. Console outputs scan status (running, succeeded, failed, skipped).
Repository Integrations	Native integrations with GitHub.com, GitHub Enterprise, Bitbucket Data Center, GitLab, and Azure DevOps. Cloud integrations provide automatic AI scanning; self-hosted versions require additional configuration.
CI/CD Pipelines	Embeds into existing CI/CD workflows for automated scanning on every commit or pull request.
IDE Integration	Real-time vulnerability detection during development for both human-generated and AI-generated code. Supports integration with AI code editors like Cursor via MCP server.
Mend Platform UI	Single web interface managing all security products. AI-specific dashboards including AI-BOM reports, Shadow AI reports, behavioral risk dashboards, and AI Insights.

5.2 Integration Methods

Mend AI currently supports models sourced from Hugging Face and Kaggle, with plans to expand. Detection covers major AI/ML frameworks (LangChain, LlamaIndex, TensorFlow, PyTorch), LLM providers (OpenAI, Anthropic, Google, AWS Bedrock, Azure OpenAI, Google Vertex AI, Azure AI Foundry, IBM watsonx.ai, DeepSeek), and model repositories (Hugging Face). The supported ecosystem continues to expand with each release.

6. Data Privacy and Security

- Mend AI-Based Code Remediation does not gather customer data for training and does not share it with third parties.
- Red teaming runs against customer applications using Mend's infrastructure with AI supplemental terms of service.
- System Prompt Hardening processes prompts to generate refinements without retaining customer prompt content.
- Mend Guardrails (runtime protection) runs entirely within the customer's application process. Prompt and response text never leave the customer's environment, therefore no text content is sent to Mend. Only structured event metadata is reported so customers maintain full and exclusive control over the data.

7. Key Differentiators

- **Code-level discovery:** Detection operates at the source code level, not at the artifact level, providing earlier and more comprehensive visibility.
- **Application-aware AI risk testing:** Red teaming evaluates AI behavior within the context of each application, not just generic model-level testing.
- **Full lifecycle coverage:** From prompt discovery and hardening to red teaming, runtime guardrails, policy enforcement, and application-level protection.
- **Proprietary scoring (AIWE):** The AI Weakness Enumeration framework provides a risk score for AI-specific security issues.
- **In-app guardrails with deeper AI visibility:** Unlike proxy-based solutions, Mend embeds guardrails directly inside the application and codebase to protect against indirect prompt injection and secure both user ↔️ agent and agent ↔️ tool interactions.
- **Automated AI governance and control:** Automatically discover prompts, harden weak prompts, and enforce approved models, providers, licenses, and AI usage policies directly in development workflows.
- **Stronger protection with lower operational risk:** In-code protection avoids latency, bottlenecks, and inline failure risks while delivering broader coverage and deeper runtime visibility than perimeter-only AI security tools.
- **Platform-native:** Mend AI can be purchased alongside Mend AppSec, including SCA, SAST, and Container security, for unified visibility and governance in one platform.

Mend.io is built for every risk, across AI and AppSec. By securing the code layer and the AI layer—and the interactions between them, where modern application risk now lives—**Mend.io** extends proven AppSec workflows to the models, prompts, and agents inside today's applications, delivering continuous protection across the entire AI application lifecycle.

Visit our website