INTRODUCING THE AI READINESS FRAMEWORK

Snyk's readiness framework for building and maturing secure AI development practices

snyk

INTRODUCTION

In 2011, Marc Andreessen famously observed that "software is eating the world," and in the nearly 15 years since, it has become widely accepted that — regardless of industry — every company is now a software company. That is, success in the modern enterprise is defined primarily by an organization's ability to build and implement software and digital workflows to bring its products and services to market.

This observation captured a revolution in software development that was already well underway by 2011: legacy, slow, waterfall-based, toilsome ways of building applications were being replaced with high-velocity, agile, automated processes and modern tech stacks. Gone were mainframes, monoliths, and cumbersome IT requisition processes. In their place came DevOps, containers, microservices, and the cloud. The story of business over the past decade and a half is largely the story of which companies embraced these trends most effectively and securely.

We are now in the midst of the next major shift in software development: artificial intelligence and machine learning. Just as every business became a software company to survive the last evolution, it's no exaggeration to say that every software organization must now become an AI company — or risk being left behind.

However, technology shifts tend to be chaotic. New capabilities and accelerators often reach rapid adoption before the full scope of associated challenges — particularly emerging risk — is understood. Security and assurance become afterthoughts in the race to adopt early, leading to a period of increased threat and uncertainty as organizations work to adapt. New technology paradigms require new security paradigms to meet the emerging need.

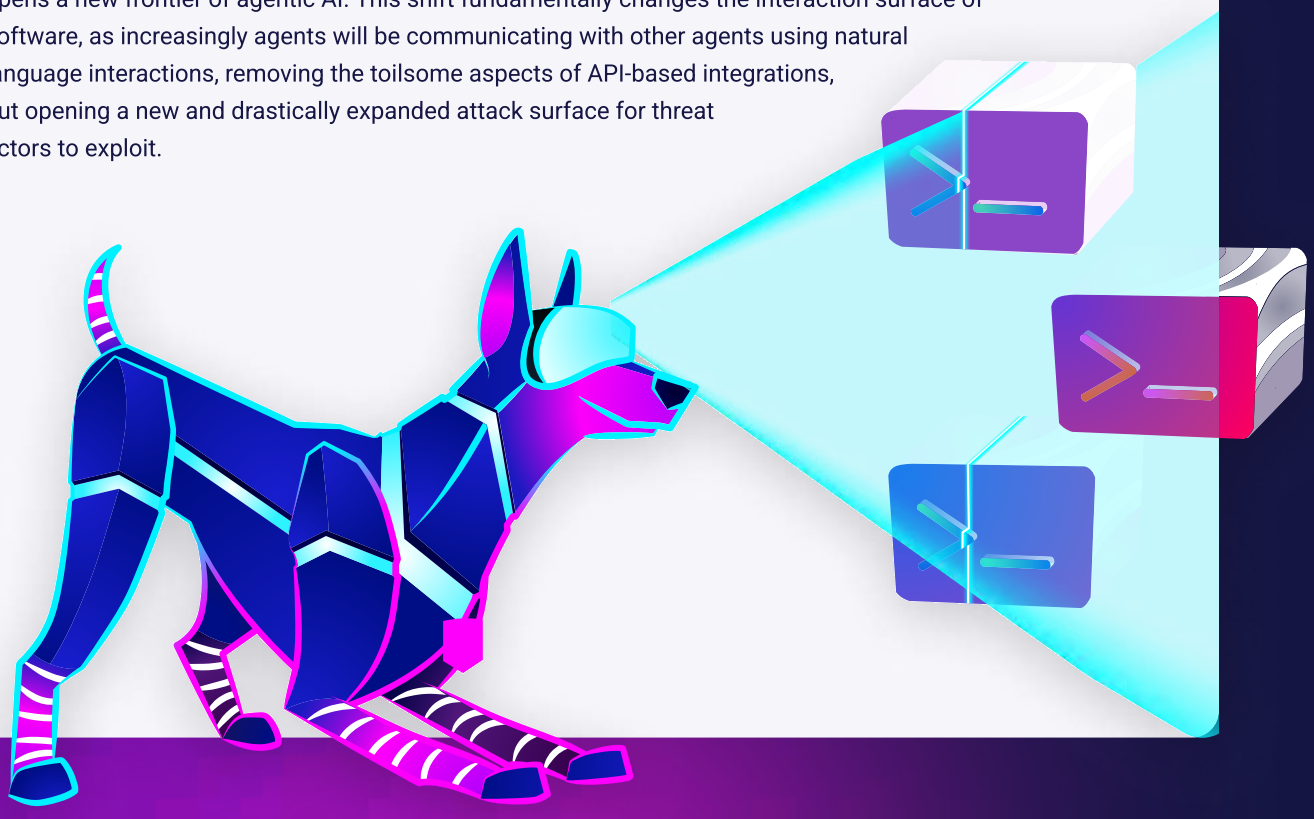
IMAGINING THE NEW AI-POWERED WORLD

Three trends are rapidly combining to fundamentally change the craft of software development as we know it:

AI is powering all software development across organizations.
Since the popularization of Large Language Models (LLMs) trained on source code in 2022, AI assistants and code generation tools have rapidly become part of every developer's toolkit. Vibe coding, in conjunction with LLMs, is now the dominant method of building. This AI-accelerated development style has already reached nearly every organization — but it carries significant risks that legacy security tools were not designed to keep up with.

Almost all software is backed by ML models and agentic services.
Moving beyond AI-accelerated development, increasingly the use and training of machine learning models has become a core aspect of software building, offering the chance to solve entirely new classes of problems and existing problems much more efficiently than traditional development. From integrating off-the-shelf LLMs to fine-tuning models, implementing Retrieval Augmented Generation (RAG), and training bespoke architectures, this AI-native development style introduces novel threat vectors that existing AppSec scanners do not detect.

Software communicates across services using autonomous, natural language interactions.
Beyond the generative aspect of LLMs, the ability of newer models to conduct goal-based reasoning opens a new frontier of agentic AI. This shift fundamentally changes the interaction surface of software, as increasingly agents will be communicating with other agents using natural language interactions, removing the toilsome aspects of API-based integrations, but opening a new and drastically expanded attack surface for threat actors to exploit.



CHALLENGES AND EVOLVING ROLES

These trends in rapid adoption of AI-accelerated and AI-native development have led to a sense of chaos within organizations, as security teams struggle to understand AI technologies from an assurance perspective, and roles within engineering teams become quickly realigned to meet new demands and opportunities. Some of the broad challenges facing early AI adopters include:

AI-generated code has a 40% vulnerability rate
According to numerous studies, LLM-based code generation tools have significant weaknesses in security. Because these models must be trained on such a large and diverse code dataset to be effective — as well as the non-deterministic nature of language models — minimizing vulnerabilities is a particularly difficult problem.

Every employee will be an App Creator
For the first time, the ability to build software no longer requires the skill set of writing code. This means potentially everyone in an organization will be able to create applications by interacting with AI surfaces in natural language, drastically increasing the number of users in the SDLC who may inadvertently inject software risk.

No consideration for authorization and integrations
As AI agents become more prevalent and begin communicating with each other to achieve goals as the primary means of software interaction, the question of non-human identity and permissioning becomes even more important.

Significantly increased attack surface
Alongside the increased prevalence of traditional software vulnerabilities introduced by genAI coding, AI-native development introduces novel threat vectors against the ML models and AI surfaces themselves, which existing AppSec tooling and DevSecOps practices are not designed to mitigate.

Compliance gaps and lack of governance
Organizations rapidly adopting AI face long-term compliance risk, as regulatory regimes are still catching up to the new problem space and forthcoming requirements are likely to be uneven and change frequently. Lack of internal AI governance enables widespread “shadow AI” adoption, hiding risks.

These challenges will not affect all stakeholder groups equally, and not in the same ways as today's software security concerns. Roles are rapidly evolving within organizations to keep pace with change, with traditional distinctions between who is and isn't a "developer" breaking down. Broadly speaking, there are three personas with responsibility for AI risk:

- **Security creates policies for AI-accelerated software teams**
- **Platform engineering implements policy as guardrails throughout the AI SDLC**
- **App creators consume the guardrails inside their daily AI-native workflows**

Enterprises must quickly develop risk assurance strategies and processes, supported by platform tooling, that provide visibility, prioritization, and policy around the new AI risk landscape to each of these stakeholder groups. But previous maturity frameworks built for earlier eras of software development do not adequately prepare organizations for the AI revolution. What's required is a new readiness framework, purpose-built for the AI era, which builds confidence and trust in AI maturity.

THE NEED FOR EVOLUTION

A decade and a half ago, the combination of agile methodologies, DevOps automation, and the arrival of cloud computing vastly increased the velocity of software development while radically changing its shape. Traditional application security, built for legacy processes and technologies, could not keep pace with the demands of modern software; in its place, DevSecOps was born, providing a roadmap for organizations to embrace the velocity of DevOps while accurately assessing and continuously improving their security governance and processes.

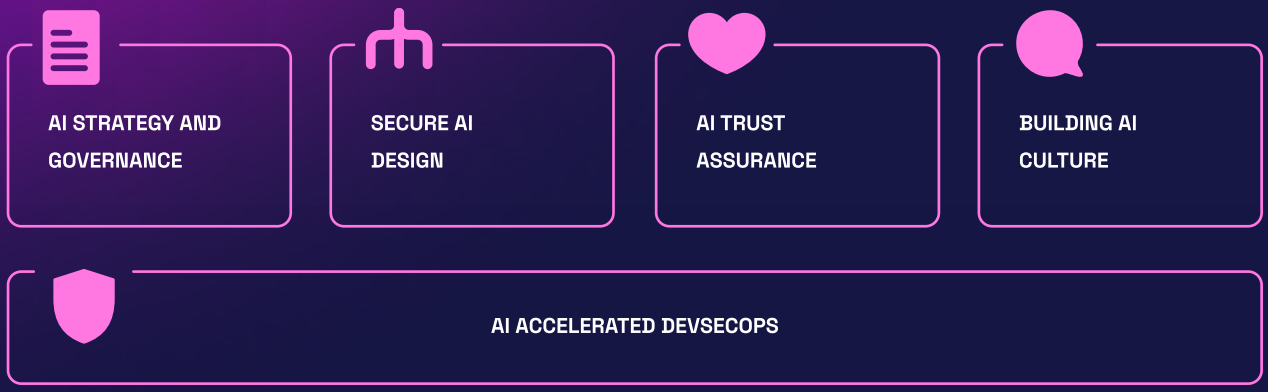
A similar revolution is currently underway. **AI TrustOps is the natural extension of DevSecOps** for modern AI-accelerated and AI-native software. It serves as scaffolding for building and maturing readiness for AI software, enabling enterprises to trust the promise of new technologies while quickly adapting to novel risks. But far from abandoning the principles of DevSecOps, AI TrustOps requires it as a foundation. AI-accelerated and AI-native software is built on top of modern DevOps. Therefore, the prerequisite to AI readiness is mature DevSecOps practice.

Thoughtfully implemented, the AI Readiness Framework enables organizations to build trust in:

- **Enhanced security posture -** Address the unique security challenges of AI-accelerated software.
- **Reduced risk of AI-specific attacks** - Proactively mitigate novel vulnerabilities targeting AI-native applications.
- **Improved data privacy** - Ensure responsible and secure handling of data used in integrated AI.
- **Increased trust and confidence -** Foster trust by ensuring secure operation of AI features.
- **Regulatory compliance** - Navigate the emerging landscape of AI regulations.

FIVE PILLARS OF AI READINESS

As organizations begin to approach the question of readiness for AI-accelerated and AI-native development, there are five key areas of strategic focus, with four pillars focused specifically on novel considerations brought about by the AI revolution:



AI strategy and governance
Establish the initial accountability model, well-documented risk posture, buy-in, and shared awareness needed to assess AI initiatives holistically.

Secure AI design
Consider AI-native risk indicators like explainability, bias, and hallucinations as an integral part of intelligent systems architecture.

AI trust assurance
Implement meaningful AI-aware risk analysis at multiple points in the accelerated DevSecOps cycle, rather than as a one-time bolted-on review.

Building AI culture
Develop the educational habits, transparency, and organizational trust necessary to establish psychological safety around AI adoption.

These first four capability pillars, while designed to address the delta between current ways of working and the paradigm shifts of AI readiness, are necessary but insufficient without their underlying foundation. Because AI TrustOps is descended from modern DevSecOps, immature organizations will struggle to become AI-ready without reliable and assured security processes. As a result, AI Accelerated DevSecOps is the most critical prerequisite to building AI Trust.

THE FOUNDATION: AI ACCELERATED DEVSECOPS

Even before the rise of AI, most enterprises were already on the path to maturing their secure software development practices. The core principles of DevSecOps remain essential, and many of its foundational capabilities are further emphasized by AI-accelerated development, such as machine-generated code. Fortunately, AI also offers ways to increase the value DevSecOps provides.

To prepare for AI adoption, most of the following signs of mature DevSecOps should be in place — with AI tools helping accelerate progress where possible:

- **Automation foundations**
  Software is built in reliable, repeatable ways. Engineering teams embrace a shared responsibility model in which developers self-service the packaging, deployment, infrastructure, and monitoring of applications they build. An everything-as-code rule ensures immutability of assets and avoids manual or "click-ops" tasks. Applications are loosely coupled rather than monolithic. Continuous integration and deployment pipelines are reliable. Metrics and data are collected for continuous improvement.

- **DevSecOps strategy and culture**
  Security objectives are explicit, shared, and clearly understood. Incentives across teams are aligned to balance risk mitigation with velocity. Ownership and accountability for application security are well defined. There is a resourced process for managing technical debt. Security learning is prioritized, especially for engineers, often through a security champions program.

- **Secure software design**
  Risk tolerance is identified and documented for each software asset. Security requirements are transparently shared with engineering teams during the software design phase. Software asset inventories are kept up to date and accessible via automated processes. Threat modeling is conducted on new software assets and is frequently updated. The application technology stack is fit for purpose and standardized across the engineering group.

- **Testing and monitoring**
  Security scanning tools are shifted left into developer workflows (such as IDEs and PR checks). There is widespread adoption of AppSec tooling and broad coverage across assets. Findings accuracy is high and regularly validated. Vulnerabilities are prioritized based on multiple sources of relevant context. Discovered issues are actionable for engineers. Notifications and alerts are relevant and proactive.

- **Response and remediation**
  There is a reliable and well-documented process for vulnerability management, including a single source of truth for issue tracking and related workflows, especially remediation. Incident response runbooks are regularly tested and updated. Frequent gamedays and tabletop exercises test novel scenarios. Postmortems are conducted following incidents, with accountable follow-up, and processes regularly updated to address gaps.

- **Analysis and governance**
  Telemetry across the software estate and relevant security tools is gathered in a single data lake. KPIs are tracked against shared and well-understood security goals. Reports for leadership are regularly generated against SLOs and identify historical trends. Policies and standards are documented and regularly updated. Security communication is transparent and blameless. Compliance checks are automated and continuously improved.

Only when maturity has been achieved in AI-accelerated DevSecOps can the four additional pillars of AI TrustOps be fully implemented. Building on a strong foundation and progressing on the path to AI readiness, organizations can start to implement new guidelines such as:

AI STRATEGY AND GOVERNANCE

GOALS AND GUARDRAILS ALIGNED WITH AI BUSINESS OBJECTIVES

Establish the initial accountability model, well-documented risk posture, buy-in, and shared awareness needed to assess AI initiatives holistically.

- **Vision and goals**
  Well-defined, shared milestones for the organization's use of AI, with specificity around planned investments and expected future value, should be the source of truth for the planning approach and adoption.

- **Governance team**
  A cross-functional center of excellence for AI should be created as an interim governing body, including stakeholders across engineering, security, and operations teams, working under an explicit shared responsibility model.

- **Policies, standards, validation**
  Following the stated goals, the overall AI risk posture should be documented and shared, linked to specific guardrails that are frequently tested against the evolving reality and updated to address the risk landscape.

- **Regulatory and compliance**
  Current and potential future compliance risk should be understood, with dynamic planning for audits and reporting, especially as the regulatory landscape around AI is still being formalized.

SECURE AI DESIGN

ASSESSING THE INTELLIGENT MACHINE RISK LANDSCAPE

Consider AI-native risk indicators like explainability, bias, and hallucinations as an integral part of intelligent systems architecture.

- **Risk definition and management**
  Perceived AI threats should align with software purpose and AI integration strategy (for example, using off-the-shelf LLMs vs. training bespoke models), considering all components and functionality as a complex whole.

- **AI/ML asset inventory**
  All machine learning models, related libraries, datasets, and AI-related codebases should be proactively identified, tagged, and stored in the software asset management system of record and made available for automation.

- **Threat and attack vector modeling**
  As new software is designed, consider novel AI/ML vulnerabilities as part of “train vs. code” decisions. Evaluate the business value of AI surfaces against the new threat landscape, and plan for resilience in the event of AI-related incidents.

- **Data integrity by design**
  Conduct extensive pre-training checks for provenance and governance of all datasets to be used in ML modeling. Develop a minimum viable scope and restrict data access for training purposes to only fully validated sources.

AI TRUST ASSURANCE

FIND, PRIORITIZE, AND FIX ISSUES IN AI-NATIVE SOFTWARE

Implement meaningful AI-aware risk analysis at multiple points in the accelerated DevSecOps cycle, rather than as a one-time, bolted-on review.

- **Bill of materials and AI-SPM**
  Continually update the AI BOM associated with all software assets. Integrate AI security analysis engines with existing aggregation tiers, surfaces, and posture management tools to develop an overall measurement of AI-accelerated risk.

- **Model security and drift detection**
  Continually test ML models for access, potential modification, and anomalies. Provide security assurance around retraining and fine-tuning operations to ensure models remain immutable..

- **Data standards enforcement**
  Continually enforce defined standards for bias detection, privacy, legality, and minimum viable scope in all datasets interacting with ML models as part of automated AI SDLC processes.

- **Model red teaming**
  Conduct frequent, recurring exploit testing of all models, both foundational and internally trained, using continually updated exploit payloads and recent AI threat intelligence from multiple trusted sources.

- **ML platform security**
  Consider the software supply chain of all tools used in model training, fine tuning, RAG, agent development, and user interaction surfaces.

- **Access controls**
  Build a flexible model for human and non-human identity and authorization that accounts for the proliferation of AI agents as well as agent-to-service, agent-to-model, and agent-to-agent communications.

- **Performance and resiliency**
  Conduct regular benchmarking against defined application performance KPIs and frequent scenario-based exercises to ensure that AI assets do not create single points of failure.

- **Incident response**
  Develop iterative runbooks for a broad range of AI-related security events. Continually update them using threat intelligence from multiple trusted sources.

BUILDING AI CULTURE

UNLEASHING AI PRODUCTIVITY REQUIRES INVESTING IN PEOPLE

Develop the educational habits, transparency, and organizational trust necessary to establish psychological safety around AI adoption.

- **Learning mindset**
  Develop an organization-wide AI risk awareness campaign with tailored training based on persona, frequently revisited to surface novel threats. Encourage a culture of knowledge sharing and continuous AI education.

- **Ownership and accountability**
  Based on existing DevSecOps best practices, create a well-understood model for AI security responsibilities throughout the organization, with attestation and auditability.

- **Collaboration and communication**
  Create cross-functional teams to collaborate transparently on the organization's evolving AI risk posture. Encourage the open sharing of concerns, and communicate clearly about identified threats and responses.

- **Adoption policies and procedures**
  Build a set of clearly documented, accessible rules of engagement to gate access to AI software tools relevant to role.

FROM DEVSECOPS TO AI TRUSTOPS

Snyk has been the industry leader in developer security and DevSecOps since 2016, helping organizations achieve maturity in modern software assurance through our developer-first AppSec platform and advisory services. Our mission has always been to create trust in the relationship between the builders of software and the risks of that software.

That mission hasn't changed, but as we have seen, the AI revolution has significantly changed both the roles of software builders and the landscape of risk that organizations must continually identify, validate, prioritize, and remediate as part of a complex development lifecycle. Just as DevSecOps answered the call in providing a roadmap for tackling the last technology revolution, AI TrustOps provides a structure for assessing and maturing your readiness for AI-accelerated and AI-native software.

Unleashing AI productivity begins with trust.

AI trust begins with Snyk.

Want to learn more about how Snyk builds trust in AI software?

EXPLORE SNYK NOW.

snyk