LEARNING MADE EASY

Harness Special Edition

AI Security

for dummies®

A Wiley Brand



Identify hidden AI security risks

Adopt AI-specific testing methods

Build a mature AI security model

Brought to you by

harness

Audrey O'Shea

About Harness

Harness provides complete application security on the industry's leading Agentic DevSecOps platform. Combining AI Security with Web Application and API Protection (WAAP) and Application Security Testing (AST), Harness helps organizations secure everything after code — including the next generation of agentic applications. Pipeline-native security can keep up with development, scale across 1000s of applications, and continuously monitor runtime risk — for both traditional and agentic applications. Security, DevSecOps, and platform teams rely on Harness to eliminate blind spots, stop advanced threats, simplify compliance, and accelerate secure delivery without slowing innovation.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.



AI Security

Harness Special Edition

by Audrey O'Shea

for dummies®
A Wiley Brand

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

AI Security For Dummies®, Harness Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

Copyright © 2027 by John Wiley & Sons, Inc., Hoboken, New Jersey. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, The Dummies Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.dummies.com/custom-solutions. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN 978-1-394-45701-4 (pbk); ISBN 978-1-394-45702-1 (ebk); ISBN 978-1-394-45703-8 (ebk)

Publisher's Acknowledgments

Editor: Elizabeth Kuball
Acquisitions Editor: Traci Martin
Senior Managing Editor: Rev Mengle
Client Account Manager: Jeremith Coward
Content Refinement Specialist: Umeshkumar Rajasekhar

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Introduction

Your biggest artificial intelligence (AI) security risk is the one you don't know about.

Consider each of the following:

- **The core problem:** Agents and AI components are proliferating faster than security teams can surface them. Developers are creating agents and pulling in large language models (LLMs), application programming interfaces (APIs), agents, and open-source models without telling security.
- **The uncomfortable truth:** Most AppSec programs have no reliable visibility into all the agents and AI assets running in their environment right now. As a result, their AI inventories are, at best, incomplete; at worst, misleading.
- **The three-part answer:** Discover what's there, test for AI-specific risks, and protect at runtime.

Agents and AI applications have different risks, different components, and different architecture from deterministic systems. The prevalence of AI and agents means your attack surface is expanding rapidly and you'll need to consider how to change the way you approach application security.

About This Book

AI Security For Dummies, Harness Special Edition, is for AppSec engineers and security practitioners who need a practical, no-nonsense framework for securing agents, AI applications, and assets.

This book covers:

- Why agents and AI applications change your attack surface
- How to find the AI in your environment, including shadow AI
- AI's inherent security dangers and how to test for them
- A simple model for running a mature agentic security program

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Icons Used in This Book

This book displays icons in the margin to alert you to key information. Here's a quick guide to the icons:



The Tip icon points out ideas and helpful information that can save you time or avoid frustration.



The Remember icon highlights key information that you'll want to store in your memory for future use.



The Technical Stuff icon flags information you can skip if you're short on time.



The Warning icon point out problems to be aware of or precautions you need to take.

Beyond the Book

This book introduces AI application security, but for an even deeper understanding of AI application security, take advantage of the following resources:

- www.harness.com
- www.harness.io/products/security-testing-agent
- www.harness.io/products/runtime-protection-agent
- https://securitybrief.news/story/harness-unveils-ai-security-coding-tools-for-devsecops
- https://owasp.org/www-project-top-10-for-large-language-model-applications/

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

IN THIS CHAPTER

- Deciphering agentic- and AI-specific components
- Examining why agentic and AI risks are invisible to traditional tools
- Delineating agentic- and AI-specific risks and mitigations

Chapter 1
Understanding AI's Effects on Your Attack Surface

Customer service chatbots have become ubiquitous. But the power of artificial intelligence (AI) is so much more, regardless of the vertical market your organization occupies. Its power is both game-changing and alarming — particularly with the rise of agents that are executing independently and often without oversight.

Organizations are in a rush to take advantage of AI's power, even as security professionals struggle to keep pace with and control of AI application access to large language models (LLMs), personally identifiable information (PII), and other privileged data. Considering the new vulnerabilities and responsibilities AI applications and agents present, developers and security professionals must work together to mitigate the possibility of attack.

Comparing the components of modern AI applications to traditional applications is a great place to start understanding those vulnerabilities.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

TERMS TO KEEP IN MIND

Keep these terms in mind as you read this chapter:

- **Agents** are AI applications that take autonomous actions like calling APIs, writing code, and sending messages. They don't just respond to specific requests; they can reason and take actions you may not have directly asked for.
- **Tools** are defined functions that an agent can invoke.
- **Model Context Protocol (MCP)** is an emerging standard that defines a structured way for AI models to connect to tools and data. It's essentially a plugin system for AI, and you should assume that any mature AI application's architecture will include MCP servers.
- **Large language models (LLMs)** analyze the context of words in massive amounts of data, so they learn to mimic human reasoning and create output including emails and code. They're in the background of those seemingly human chatbots we're all familiar with, and much more.
- **Retrieval augmented generation (RAG)** is how applications give models access to your internal data at query time, pulling relevant content from a vector database and injecting it into the prompt.

Understanding Modern AI Application Anatomy

Foundations for an application are the same whether it's a traditional application or an AI application. Code, dependencies, databases, and application programming interfaces (APIs) are the foundations of both traditional and AI applications.

However, AI applications have an entirely new set of components that will keep you up at night as you update your mental model of what an application is. When you're talking about AI components, you're discussing LLMs, embedding models, vector databases, AI APIs, agent frameworks, and Model Context Protocol (MCP) servers.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Following are the AI application components and what is found or working in each:

- **Model:** This layer holds the LLM or AI model, whether it's hosted by a third party or self-hosted.
- **Orchestration:** The coordination of the AI components and custom logic are found here.
- **RAG:** Vector databases and document stores are accessed at this layer.
- **Tools and integrations:** Endpoints, authentication, and third-party services that AI calls are in this component.
- **MCP:** Here you'll find servers, tools, resources, and prompts that create the connection between an agent and an LLM.



An AI model is the control center that performs the magic math behind the scenes to sort and filter data, make decisions, generate content, and make predictions to create the desired output. An AI application is the whole package that provides the user interface (UI) that enables the user to make requests to and receive responses from the AI model. LLMs are a popular example.

Unveiling AI Problems and Risks

As enterprises rush to incorporate agentic AI, security professionals are struggling to assess and mitigate the risks. Agentic AI is being implemented so fast that standards for developing these tools are not yet in place.

Although you can use traditional security to limit damage in deterministic apps, remember that AI's reach is often deeper into your data and works at processor speed, not human speed. It has risks that traditional security wasn't designed to find or stop. Take another look at AI's layers, this time from the security point of view, and the questions you should ask at each layer:

- **Model:** Ask yourself, where did the LLM or AI model come from and what data does it touch?
- **Orchestration:** Who controls the flow, decides which interactions are allowed between different components, and

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

decides what can be manipulated? What's protecting AI agents from prompt injections?

- **RAG:** With RAG linking public models to your secure internal networks, ask yourself what data can be retrieved and by whom?
- **Tools and agents:** What is the blast radius of your AI model? How extensive will the damage be if it is breached?
- **MCP:** The pieces that live at this layer define how an AI model interacts with data and what actions it takes, and can provide predefined message templates, call specific AI application behavior, or initiate predefined workflows. What is exposed from outside your perimeter?
- **API/integration:** What endpoints can reach the model? What APIs is AI using and connecting to? What APIs are increasing your attack surface? What information leaves your system? Finding AI applications that already exist in an environment is difficult because they're largely shadow AI, often spun up without IT department approvals. Configuration management database (CMDB) and other software inventory tools are blind to AI because unlike traditional applications, AI often runs as an API, uses cloud platforms, and often runs code in cloud containers that disappear within minutes.



REMEMBER

New vulnerabilities require new tools to manage them and mitigate their impact. What's your plan to address security blind spots in your AI?

Traditional AppSec tools have blind spots when applied to AI because they were built to analyze code and known dependencies, not to reason about a model that can be manipulated through natural language to behave in ways its developers never intended.

Perhaps the biggest problem with rapid implementation of agentic AI is that much of it is shadow AI, and you can't secure what you can't see. For more information about finding AI in your environment, check out Chapter 2.



WARNING

The Harness 2025 survey revealed that AI-native apps are already under threat; 76 percent of respondents have already experienced an LLM prompt injection incident, 66 percent have experienced an LLM code incident, and 65 percent have experienced LLM jailbreaking.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Pinpointing the big ten risks and mitigations for LLM applications

The Open Worldwide Application Security Project (OWASP) lists the top ten risks for LLM applications and gives some direction on how to mitigate those risks. Here's a quick summary of the OWASP Top 10 list in order. The mitigations listed here are a starting point, not an exhaustive list.

Prompt injection

Prompt injection damage can be mitigated, in part, by ensuring that output formats are defined and input validation is required, filtering input and output and requiring a human being to approve high-risk actions.

Sensitive information disclosure

This can happen at any stage of development and deployment. From the beginning, ensure that sensitive information in the training source is scrubbed or masked before the model is trained. If possible, instead of outputting specific values, use ranges. For example, instead of an output of $125, use "between $100 and $200." Then it's important to make sure there are guardrails to prevent data being leaked in response to a prompt, for example.

Excessive agency

Employ the principle of least privilege. If an LLM doesn't need to access broad functions, then block it from using them. Ensure that they have permissions and access only to the extensions they need to do the job and nothing more. Add a human-in-the-loop on any action that could have an adverse impact so they can approve it before the email is sent or the action is done.

Supply chain

Ensure that their security is sufficient and complies with all relevant security requirements. Red team their systems that will touch yours.

Data and model poisoning

Know where your data comes from by vetting sources. Make sure your security will keep your model from accessing unauthorized data sources. Use software to monitor data changes, and red team it regularly.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Unbounded consumption

When inputs are unbounded, LLMs can prompt excessive use of resources and lead to denial of service (DoS) problems, excessive cost related to usage charges, service degradation, and even unauthorized replication of the model. Establishing user quotas and setting timeouts and process throttling will help avoid excessive resource consumption. Access controls using the principle of least privilege and role-based access control (RBAC) will help here, too. LLMs can also be trained to identify and stop queries that appear to be malicious.

Misinformation

Even if data hasn't been poisoned, LLMs and agents can produce information that's wrong. Use RAG to retrieve information that is verified and relevant. Use tools to validate outputs. Encourage human oversight to verify outputs against known reliable sources. Ensure that you've communicated the risks and limitations of generated content to users.

Hidden context exposure

Ensure that anything in a system prompt, developer instructions, retrieved policy text, or schemas of tools and functions doesn't expose sensitive information including filtering criteria, internal rules, API keys, credentials, or other sensitive information. Even something that seems innocuous might give clues to a savvy attacker. Application developers should ensure that hidden context has little or no direct security impact.

Vector and embedding weaknesses

Ensure that data is sanitized and PII is removed before it's incorporated into the embedding model. Any combined data should be reviewed to ensure that access levels are adequate and mismatch errors are prevented or addressed. Ensure that only vetted sources of information are used. Ensure that adequate, fine-grained access controls are employed.

Improper output handling

Remember to use a zero-trust approach and require input validation. Ensure that you have monitoring systems that can detect and alert you or take action against attempted breaches.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

OWASP Top 10 for LLM Applications 2026, adapted from the original, used under CC BY-SA 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/deed.en).


TIP

Download the complete OWASP Top 10 for LLM Applications 2026 at https://genai.owasp.org/resource/owasp-genai-llm-top-10/2026/.

Recognizing sources of AI sprawl

A 2025 Harness survey of 500 security professionals and decision-makers across multiple countries revealed that 62 percent of security practitioners have no way to tell where LLMs are in use across their organization. Let that sink in.


WARNING

If you can't see every AI model, API, and agent framework running in your environment, you aren't managing your AI risk effectively.

It's difficult to quantify AI's effect on your attack surface when agents and AI components are entering your environment rapidly and without your control. Here's how AI sprawl can happen:

- Developers reach for Anthropic, Hugging Face, LangChain, OpenAI, and dozens of other providers without security review.
- Agents are exposed to the internet and allowed to connect to third-party services without appropriate supervision.
- Shadow AI is the new shadow IT, occurring where AI apps, integrations, and agents that your team has never seen are already in production.
- Open-source models are downloaded and self-hosted with no provenance checks.


REMEMBER

To reduce the risk of shadow AI and AI sprawl, it's important for security teams and development teams to work together from the start.

Listing risks that require an inventory

Risk management in cybersecurity requires identifying, analyzing, assessing, and communicating risks and deciding how to prioritize and handle each risk.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

The risks you can't manage without an inventory include the following:

- Models processing sensitive data without data handling reviews
- Third-party AI APIs with unknown retention and training policies
- Vulnerable or compromised open-source models in your artifact pipeline
- AI agents operating with excessive privilege

Ditching Traditional SBOMs for AIBOMs

AI bills of materials (AIBOMs) extend far beyond the traditional software bill of materials (SBOM). They're a machine-readable inventory of all your AI assets. An AIBOM identifies and documents model lineage and provenance, training datasets and data sources, model risks and vulnerabilities, and dependencies and third-party components. An AIBOM provides reliable information enabling easier compliance and mitigating risks associated with AI.

For more about AIBOMs, see Chapter 2.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

IN THIS CHAPTER

- Refining AI discovery
- Prioritizing risks
- Describing agents and their components
- Examining the contents of an AIBOM

Chapter 2
Discovering AI in Your Environment

The foundation of artificial intelligence (AI) security is discovery. You can't test or protect what you don't know is there, so discovery is essential. Here are a few of the challenges in AI discovery:

- Applications are usually hosted.
- AI is transient in nature.
- Shadow AI is ubiquitous.



TIP

For more information about the structure of AI, its components, their function, and shadow AI, see Chapter 1.

Unlike traditional application security, AI-native security requires you to dig deeper and continuously to find and secure large language models (LLMs), Model Context Protocol (MCP) servers, and generative AI (GenAI) services.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

AI Discovery in Practice

Conventional security tools are static in nature and focus on code vulnerabilities, but AI functions in a more transient way, and most AI vulnerabilities can't be detected with simple code analysis. AI security tools must find vulnerabilities in real time, before their evidence disappears.

Monitoring runtime traffic provides a way to see every LLM, MCP server, and third-party GenAI service that's communicating with your environment. It provides visibility into AI application programming interfaces (APIs) as they're working. AI security must capture data flows and model behavior at runtime to detect and respond to AI's unique threats.

Today's security approach must include better scanning, identifying AI endpoints, and finding shadow AI. AI's inherent threats that traditional code analysis prior to deployment will miss are:

- Prompt injections
- LLM jailbreaking
- Sensitive data leaks through AI APIs

Securing LLM models and applications requires the ability to see into every API connection involving the model and continuously monitoring inputs and outputs. AI's transient nature also requires security that can see and block damaging activity as it's happening.

SHADOW AI IS A PROBLEM AT SCALE

According to Harness's 2025 survey of 500 security practitioners and decision-makers across the United States, United Kingdom, France, and Germany, 70 percent of respondents say "it seems like a new API connects an LLM to sensitive data every day in their organization." And 72 percent identify shadow AI as "a gaping chasm in their security posture."

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Scanning codebases

Three potential sources of attack that codebases must be scanned for are:

- AI library imports
- API calls
- Model references

All three expand the attack surface, so it's imperative to determine when and where they happen.

Here are the possible problems with library imports:

- **Importing a misspelled library:** Hackers can hide malicious code in misspelled libraries that triggers automatically when the library is uploaded.
- **Unpinned versions:** Code that installs the latest library versions automatically can potentially wreak havoc with software dependencies.
- **Supply chain attacks:** Attackers infiltrate package maintainers, enabling malicious code to be installed via updates.

Code that calls APIs merely by description can result in unpredictable results. The rapidly increasing number of AI-driven API calls make public APIs desirable targets for malicious actors.

Unpinned model references can open your AI to version instability by allowing AI to access unexpected model changes.

Identifying endpoints

AI applications access LLMs through APIs, making AI largely API driven. Identifying those APIs, their location, and what data of yours they may share is essential to securing your AI application. If the APIs aren't secure, then your AI application isn't secure either.

APIs that access unsecured LLMs can possibly subject your AI to data leaks and prompt injections. Compliance requires that all your LLM integrations are identified. Documenting LLM integrations and APIs is essential in risk management and data traceability.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Bringing shadow AI to light



REMEMBER

Shadow AI is the rule, not the exception. Most security teams are shocked by what surfaces with discovery's first scan.

Shadow AI refers to AI deployments and integrations that exist in production but never went through security review and approval. Shadow AI is quickly outpacing shadow IT as the biggest problem in IT security.

Part of the problem is that employees can initiate shadow AI without even realizing it through installing a seemingly harmless browser extension or using chatbots and other AI tools. Often it's their quest for efficiency that results in shadow AI.

To make finding shadow AI even more difficult, agentic AI works autonomously, initiating connections and choosing tools on their own to answer queries without human oversight. For agentic and shadow AI, sophisticated scanners are a necessity, not a luxury.



TIP

One option for finding shadow AI in your environment is Harness's AI Posture capability which provides automatic discovery of all AI assets across models, APIs, and MCP servers.

Classifying Discoveries and Prioritizing Risks

Certain data regarding each AI and agent component in your inventory must be recorded, enabling you to sort and determine security priorities for those items. For each component in your inventory, capture the following:

- Model type/provider identity and version
- Data inputs and outputs, especially anything touching personally identifiable information (PII) or sensitive internal data
- Data sensitivity
- Access and permissions (what can this model or agent do)
- Owner and deployment context
- Any vulnerabilities discovered in testing

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

- Adherence to standards like the Open Worldwide Application Security Project (OWASP) LLM Top 10 to avoid known vulnerabilities

High-risk signals in your AI inventory include the following:

- User-controlled inputs
- Access to sensitive data
- Autonomous action capabilities
- Third-party model providers
- Connections to external inputs and outputs

Identifying Agents and Their Building Blocks

Agents and AI components are highly interdependent. AI assets call each other, share data, and take action. For example, agents can call APIs and interact with:

- Your data
- MCP servers
- Third-party LLMs
- MCP clients

They're very powerful, and potentially dangerous. It's up to you as the security guru to keep them under control. Agents work by employing:

- **Autonomy:** Yes, they can work and solve problems without human oversight, but they have limits called *guardrails.*
- **A reasoning engine:** This is usually an LLM and is the brains of the process.
- **Memory:** Agents can use short- and long-term memory.
- **APIs:** Agents use APIs to retrieve information they want that's outside of their training data. APIs are also used to enable the agent to perform actions.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.



REMEMBER

Discovery in an AI environment is a capability, not a one-and-done project. AI agents and components are being rapidly added to your environment and it's essential that your inventory keeps pace.

Building Your AIBOMs

A typical software bill of materials (SBOM) is a structured, machine-readable document that inventories aspects of all software including:

- Dependency lists
- Suppliers
- Open-source libraries
- Package managers

However, SBOMs are inadequate when working with AI. In addition to SBOM information, an AI bill of materials (AIBOM) includes the following:

- Model provenance
- Data lineage and training dataset
- Vendor trust
- Runtime behavior

Expect the list to grow as AI becomes even more powerful in the future and agentic AI's use is briskly, exponentially expanded.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

IN THIS CHAPTER

- Explaining why AST tools fail AI
- Identifying AI-specific vulnerabilities
- Integrating AI into your CI/CD pipeline
- Using AI firewalls

Chapter 3
Testing and Protecting AI Applications

Application security testing (AST) tools were designed in a world without artificial intelligence (AI). In a world where AI-native tools are increasing by leaps and bounds, we need a new approach and different, AI-specific tools.

Examining Why AI Testing Requires More Than AST

AST tools alone aren't enough to secure AI-native applications. They can analyze code, dependencies, and known attack patterns. What they can't do is tell how an AI agent will behave after it's built and deployed, based on scanning or analyzing static code.

AI's power and its danger lie in its ability to change what it's doing based on human language input and using inference to determine what action to take next. AI is making humanlike decisions in many cases without human oversight.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Consider the usefulness and danger in these autonomous AI decisions:

- **Financial decisions:** AI can be used to automatically approve or deny a debit card purchase or ATM transaction and execute a stock trade. It can also be used to detect fraud in financial systems. The danger is that the AI begins to make decisions about approvals or fraud that are wrong and cause harm to consumers.
- **Vehicle control:** Modern vehicles use sensors to control speed, apply brakes, and map directions to your destination; in fully autonomous situations, they do so without human intervention. Poorly controlled sensors can cause minor and major auto accidents.
- **Chat interfaces:** Chat interfaces can save work and time, but they also can be vulnerable to data leakage due to prompt injection attacks.
- **Cybersecurity:** AI can apply security patches, detect traffic anomalies, predict risks, and automatically trigger incident response. Danger lies in AI making wrong decisions or alerting on the wrong incidents.
- **Healthcare:** Autonomous assistants can provide support such as patient engagement and verifying patient identity, assist with diagnosis and surgery, and analyze health records. Significant harm is posed to patients if those assistants are open to attack because of vulnerabilities that weren't caught in development, or if they send patient information to an unauthorized location.

AI is largely application programming interface (API) driven. AI-specific tools are designed to protect a system against malicious code that is launched when an API is called. AST tools are not.

AST tools also can't predict whether the large language model (LLM) will be able to determine whether human input should be followed or rejected.

Common AST tools include the following:

- **Static application security testing (SAST):** Scans code to find syntax errors, input validation rules, insecure references, hardcoded credentials, and other coding errors.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

- **Software composition analysis (SCA):** Looks for and manages open-source and third-party components within a program.
- **Dynamic application security testing (DAST):** Provides external attack simulation to identify code needing remediation based on runtime results.

Delineating the New Class of AI-Specific Vulnerabilities

With any new technology, the bad actors are quick to discover its vulnerabilities. It's important that AppSec engineers test for the new AI-specific vulnerabilities and risks, including the following:

- Prompt injection
- Inadvertent data exposure
- Model supply chain risk
- Insecure agent behavior
- Data poisoning
- Model poisoning
- Model theft
- Excessive agency


TIP

Unlike Structured Query Language (SQL) injection, which can be detected by examining adherence to syntax rules, AI prompt input uses normal human language, which can be used to trick AI reasoning and is, therefore, harder to detect compared to expected behavior by a user or other agent. Ensure that you have input validation and sanitization in place.

Using AI-Specific Testing Techniques

Unique vulnerabilities require unique ways to identify them and secure against malicious action. AI-specific testing techniques include the following:

- Adversarial prompt testing and LLM red teaming
- Output analysis and data leakage detection

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

- Model provenance verification and software bill of materials (SBOM) and AI bill of materials (AIBOM)
- API security testing for LLM endpoints

Adversarial prompt testing tries to break an AI model, so developers know what needs fixing. Adversarial prompt testing methods that may be employed include the following:

- Jailbreaking
- Boundary testing
- Prompt injection
- Prompt leaking

Output analysis and data leakage detection use a series of red-teaming inputs to test the output of the model.

Model provenance verification determines the path from the model's inception to today, including the data used, while the AIBOM is used before and after deployment to verify the supply chain's integrity.

API testing for LLM endpoints involves several components such as automated API discovery, continuous integration/continuous delivery (CI/CD) pipeline testing, and runtime protection to analyze API traffic behavior, anomalies, and block threats in real time.


TIP

Look for a solution that has AI-specific testing throughout the software development life cycle (SDLC), like the security products by Harness, specifically designed to find anomalies in generative AI (GenAI) and embedded AI models and agents.

Integrating AI Testing into Your Existing CI/CD Pipeline

If you're new to using AI agents and applications in your space, then your CI/CD pipeline needs updating too.


TIP

Most agent risks are discoverable only after the agent is built and running, but there are steps you can take to catch risks. In the CI/CD pipeline, you can scan the static skills and prompts stored in the code repository and run adversarial tests against agents in a test/development environment before deploying into production.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

AI agent and application deployment is more complex than traditional software deployment. AI includes the full stack. defining how the AI application behaves, how data is gathered, the LLMs and other models it uses, the logic, the retrieval augmented generation (RAG) used to keep data current, guardrails, orchestration code, and agents and tools.

More sophisticated gates and controls are required in AI application deployment because AI is so powerful and malleable. Version prompts, configurations, and policies should be considered as code.

Understanding Why the Old Gates Aren't Enough

Standard pipelines gate on criteria like whether it compiles, whether unit tests pass, whether it lints cleanly, and whether it passes a security scan. Those all assume deterministic code paths. An agent's code path is chosen at runtime by a model reading a prompt and some tool outputs, so the same commit can behave differently across runs. That's the core problem new gates need to address.

New gates worth adding include the following:

- **Behavioral evaluation gate:** Run a fixed suite of representative tasks (not just unit tests) and score outputs against rubrics or reference answers. Block merge if the pass rate drops below a threshold versus the baseline model/prompt version. This is the closest analog to "tests" but for behavior rather than logic.
- **Tool use/action safety gate:** Because agents can call tools (write files, hit APIs, execute code, spend money), gate on whether the agent invokes only the tools it's authorized to use in the evaluation suite, whether it respects any allow lists, and whether it avoids destructive actions on dry-run tasks. This is closer to a security scan but scoped to actions taken, not just code written.
- **Regression-on-drift gate:** Compare current outputs against a "golden" snapshot from the last approved version,

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

specifically flagging cases where the agent's plan or reasoning trace diverges significantly. This is useful for catching silent degradation when you swap a model version or tweak a system prompt.

- **Cost/latency budget gate:** Agents can spiral into long tool call loops. A gate that fails the build when token spend, tool-call count, or wall-clock time per task exceeds a set budget in the evaluation suite can catch runaway loops before they hit production.
- **Prompt injection/adversarial input gate:** Run the evaluation suite with known injection patterns embedded in tool outputs or documents the agent processes, and fail if the agent's behavior changes in unsafe ways (leaking data, ignoring instructions, and taking unauthorized actions).
- **Human-in-the-loop approval gate for high-stakes changes:** Not automated, but structurally, any change to the system prompt, tool permissions, or model version routes through a review step distinct from code review, because these often aren't "diffs" a normal code reviewer will catch the implications of.
- **Nondeterminism/variance gate:** Run the same task *n* times and check output variance stays within an acceptable band. If small prompt tweaks cause wildly different tool-call sequences, that's worth flagging even if each individual run "passes."


TIP

Most teams get the most value from starting with a behavioral evaluation gate and a cost/latency budget gate — they catch the majority of regressions cheaply. The injection and variance gates are worth adding when the agent has real users and real tool access, because that's where the failure modes get expensive.

Protecting at Runtime with the AI Firewall

Preproduction testing alone is not enough to protect your environment. Real users' inputs can be unpredictable.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Web application firewalls (WAFs) and web application and API protection (WAAP) are inadequate for AI because they aren't designed to detect AI-specific attacks like prompt injection.

SQL has a finite, recognizable syntax, but prompt injection doesn't. An attacker can enter unexpected instructions or embed malicious code in an uploaded file.

AI firewalls work like other firewalls except that they use semantic reasoning to understand not only what is being said, but what the intent of the input is.

An AI firewall's job is to:

- Inspect and filter prompts before they reach the model, blocking prompt injection in real time.
- Block outputs that contain sensitive data or toxic content.
- Provide rate limiting and anomaly detection on AI API endpoints.
- Enforce custom AI usage policies and guardrails.

Runtime findings should then initiate a feedback loop in which malicious activity the firewall catches is fed back into the discovery inventory and testing coverage.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

IN THIS CHAPTER

- Establishing governance going forward
- Recognizing the importance of continuous discovery
- Identifying roles and responsibilities
- Describing the iterative process

Chapter 4
Running a Mature Agentic Security Program

Discovery is the foundation of any artificial intelligence (AI) security program because you can't protect what you can't see. (For more on discovery, see Chapter 2.)

Testing, including red teaming, is essential to ensure that the security measures you've put in place are working. Testing tells you where you need to tweak your protection measures. Firewalls work at runtime to catch the unexpected. (See Chapter 3 for more on testing and firewalls.)

Discovery, testing, and firewalls are essential, but what holds it all together is a mature AI security program.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Establishing Governance Before Sprawl Wins

You've discovered the AI assets in your environment and taken protective measures, but you must take action to set up procedures before AI sprawl takes over again. You can achieve this by using policies, workflows, and ownership.

Create/cultivate each of the following:

- **Set an AI usage policy.** Identify and document the models and providers that are approved, what they're approved for, and what data classifications may be used with them.
- **Establish approval workflows.** Team members in your organization will want to add more AI integrations to increase efficiency, add value to customer interactions, and so on. It's important to ensure that there is an established approval process to maintain control over sprawl, and that it's lightweight enough that developers won't route around the approval process.
- **Establish ownership.** Security can't do this on its own. Determine who is accountable for each AI component in your inventory so they'll take ownership and help prevent AI abuses.

Making Discovery Continuous

Nothing is static in an AI-driven world. When AI components are added to your environment on a weekly basis, a quarterly audit is already out of date and you're playing security catch-up.



TIP

Continuous discovery can be a daunting task, so look for capabilities to help you like Harness's AI Posture.

Continuous discovery is essential and the foundation of securing AI use. Your inventory should update in real time as your environment changes. Hopefully, any changes have gone through the approval process and you're aware of them before they go live, but something as simple as an employee installing a new extension for their browser can be a source of AI sprawl. (To find out more about shadow AI, see Chapter 1.)

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Review your inventory and use it as the basis for your testing priorities. Components with the highest risk should be tested first and most often. Create a testing schedule based on those priorities and incorporate it into your process.

Building Cross-Functional Relationships

A fully functioning AI security program requires more than your security team. Your mature AI security program needs a little help from your friends.

High-performing cross-functional relationships include the following teams and responsibilities:

- **AppSec's role:** AppSec owns the program, the tooling, and the standards.
- **Business owner's role:** Sometimes the business owner or project champion takes on the developer role, because many agents aren't built by developers.
- **Developer teams' role:** Developer teams are responsible for integrating security into how they build AI, not treating it as an afterthought or an external gate.
- **Machine learning (ML) and data science teams:** This group owns the testing of the model algorithms, validating the data sources, and framing the problems the AI is solving.

Charting a Simple Process for Managing AI Application Security

There are four stages in this simple model. Remember, however, that discovery is continuous. This isn't a one-and-done project. The model for AI application security is an iterative process that includes the following four stages:

- **Stage 1 — Discovery/inventory:** At this stage, you'll use tools to assist you in searching out any AI that is running in your environment. You must know what's there before you can continue the process.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

- **Stage 2 — Testing:** Testing requires AI-specific tests that are integrated into your continuous integration/continuous delivery (CI/CD) pipeline. The old CI/CD tests are not designed to find AI-specific problems.
- **Stage 3 — Protection:** Your runtime firewall is in place across AI-exposed APIs.

  Your AI can't be secure until your APIs are secure.
- **Stage 4 — Monitor/observe:** Your program is an iterative process that includes continuous discovery, governance, cross-functional ownership, and feedback loops operating together. It must be monitored to ensure that your organization stays secure and nothing falls through the cracks.



REMEMBER



REMEMBER

Most organizations are operating at Stage 1 without full visibility. It's important to have a way to discover all AI assets in your environment.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

Chapter 5
Ten (or So) Components of an Effective Agentic Security Program

Developing an effective agentic security program can be a daunting task. To help you get started, here are ten (or so) components that you should have or put in place to secure your agentic artificial intelligence (AI):

- **Inventory:** You'll need a complete and current inventory of all AI in your environment. This includes models, Model Context Protocols (MCPs), application programming interfaces (APIs), agent frameworks, embedding services, and the like. You can't protect what you don't know is there, so without an inventory, you're just hoping for the best. Discovery is not an optional step — it's your starting line!
- **Risk assessment:** Everything AI in your environment must be evaluated and assigned a risk score. Knowing which applications have a high risk helps you manage time and resources to monitor or mitigate those risks or pull the plug where appropriate.
- **AI bill of materials (AIBOM) for each application:** You'll need an AIBOM for every AI and agentic AI in your environment. This

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

includes dependencies, suppliers, open-source libraries, package managers, model provenance, data lineage and training dataset, vendor trust, runtime behavior, and any other important information about the application.

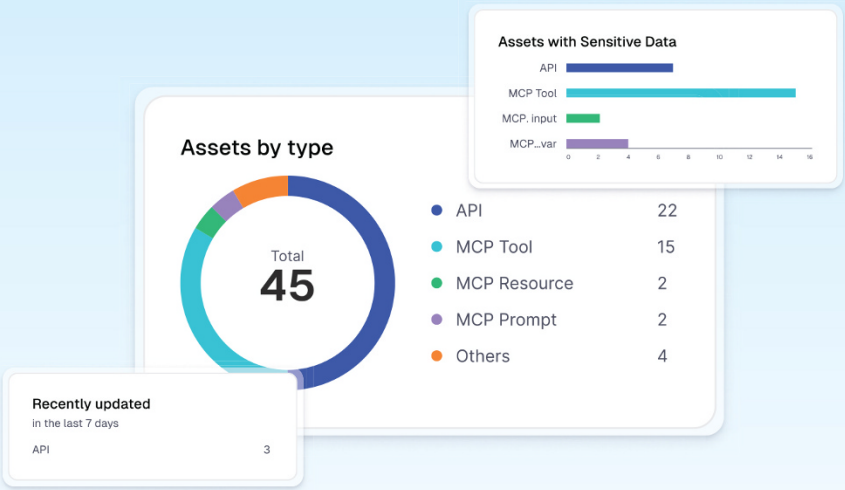
- **AI-specific testing methods and tools:** Application security testing (AST) tools were designed for a static world and aren't enough when working with AI, especially agentic AI. AI uses APIs and works in temporary spaces. You need tools built to handle the transient nature of AI, detect malicious code when an API is called, and analyze the ability of large language models (LLMs) to correctly interpret human input. (Is this person making a legitimate request?)
- **Red teaming:** Agents that can call APIs, write files, browse the web, or send communications have a real blast radius. The principle of least privilege applies here. Ensure that your red-team puts your application through its paces before attackers do.
- **Runtime AI firewalls:** Production AI applications need an AI firewall that's providing real-time inspection of prompts, outputs, and anomaly and abuse prevention. If you're relying on preproduction testing alone, you're missing the third layer.
- **Observability:** Your system must be logging, monitoring, and alerting for AI-specific behavior. Simply using infrastructure metrics won't work here. Having a system to trace every interaction, prompt, and response, and having an audit trail for incident investigation and audit reporting, is critical.
- **Cross-functional ownership:** Developers and security need to work together to ensure that agentic AI is secure, right from the beginning.
- **Governance:** Guidance such as AI usage policies and guardrails, approval workflows, and review and monitoring requirements are part of governance. Governance policies also need to include how to handle AI-specific incidents, including data leakage through the model, agent manipulation, and prompt injection at scale. Ensure that someone has answered the question "What do we do if the AI is being exploited right now?" and has documented the procedure.

These materials are © 2027 John Wiley & Sons, Inc. Any dissemination, distribution, or unauthorized use is strictly prohibited.

harness

Detect and Protect AI At Machine Speed

72% of organizations have already experienced at least one production incident directly caused by AI-generated code



Get Started Today

Sign up for an in-depth demo today to see how Harness can help secure your assets and agents.

The AppSec engineer's guide to the AI-native era

Your biggest AI security risk is the one you don't know about. AI and agents are getting deployed faster than security standards can keep up with them, so it's important that you know what you're up against, how it's different from traditional application security, and how to protect your organization against the pitfalls and malicious actors.

Inside...

- Define your new attack surface
- Uncover hidden AI and AI agents
- Update your CI/CD pipeline for AI
- Develop a mature AI security program
- Build iteration into your AI detection
- Identify the big ten risks for LLM



Audrey O'Shea is a computer consultant, former educator, and IT author with numerous industry certifications. She has written four technology books leading to IT industry certifications and has served as technical editor on several others. She holds two degrees and a New York State teaching license.

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-394-45701-4
Not For Resale



for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.