AWS Security Agent

A Practical Guide to Agentic Application Security:
What It Is, What It Solves, and How to Implement It

Covers: design review · threat modeling · code review · autonomous penetration testing · IDE & MCP integration

Prepared July 2026

BLUE LAKE IT

bluelakeit.co.uk

Executive Summary

AWS Security Agent is Amazon Web Services' agentic application-security service, now part of the broader AWS Continuum security suite. Rather than acting as a single scanning tool, it is a frontier AI agent that reasons about an application's architecture, code, and runtime behavior, and then takes action: producing threat models, reviewing pull requests, and running autonomous, exploit-validated penetration tests. It is designed to compress security work that has traditionally taken days or weeks — architecture reviews, threat modeling, and penetration testing — into an on-demand, continuous capability that keeps pace with modern development velocity.

This guide summarizes what AWS Security Agent does, the specific problems it is built to solve, its core capabilities, and a step-by-step path to implementing it across the design, development, and deployment stages of the software lifecycle.

At a Glance

Category	Details
What it is	An AI-driven, agentic application security service (part of AWS Continuum)
Core capabilities	Design review, STRIDE-based threat modeling, full-repo & PR code review, on-demand penetration testing
Status (as of mid-2026)	Penetration testing: GA · Code review & threat modeling: Preview
Where it fits	Design time → development time → deployment time, in one unified offering
Integrates with	GitHub, GitLab, Bitbucket, Confluence, Kiro, Claude Code, and any MCP-compatible IDE
Regions (pentesting GA)	US East (N. Virginia), US West (Oregon), Europe (Ireland, Frankfurt), Asia Pacific (Sydney, Tokyo)

1. What Is AWS Security Agent?

AWS Security Agent was previewed at re:Invent 2025 and has since expanded through 2026 into a full application-security offering. AWS classifies it as a **frontier agent**: an autonomous system that works independently toward a goal, scales to handle many tasks concurrently, and can run for hours or days without constant human supervision — a meaningfully different operating model from a conventional AI assistant that responds to individual prompts.

In practice, the agent ingests an application's design documents, source code, and architecture context, builds an internal understanding of how the system is put together (its components, data flows, and trust boundaries), and then uses that understanding to perform four connected functions:

- **Design review** — validates architecture and design documents against security requirements and compliance frameworks before a line of code is written.

- **Threat modeling** — automatically generates STRIDE-based threat models from design docs or source code, mapping components, threat actors, and attack vectors.
- **Code review** — performs reasoning-based analysis (not just pattern matching) across pull requests or entire repositories, producing fix commits and remediation guidance.
- **Penetration testing** — autonomously discovers, exploits, and validates vulnerabilities in deployed applications, chaining findings the way a human tester would.

Because these four functions share the same underlying reasoning about the application, findings from one stage inform the others — for example, a threat model can highlight the trust boundaries that a subsequent penetration test should target.

Why AWS Built It

AWS has framed Security Agent partly as a response to the pace at which frontier models — including Anthropic's Claude Mythos and Claude Fable — can now be used to discover software vulnerabilities. AWS security leadership has described this shift as raising the bar for how fast real attackers (and defenders) can find weaknesses, which is part of the motivation for building matching AI-native defensive tooling.

Architecture Overview

At a high level, AWS Security Agent sits between an application's sources of truth and the surfaces where developers and security teams actually work. It pulls context from source code, design documents, and the running application; reasons about that context as a single agent; and pushes results back out to the console, IDEs, and existing Git workflows.

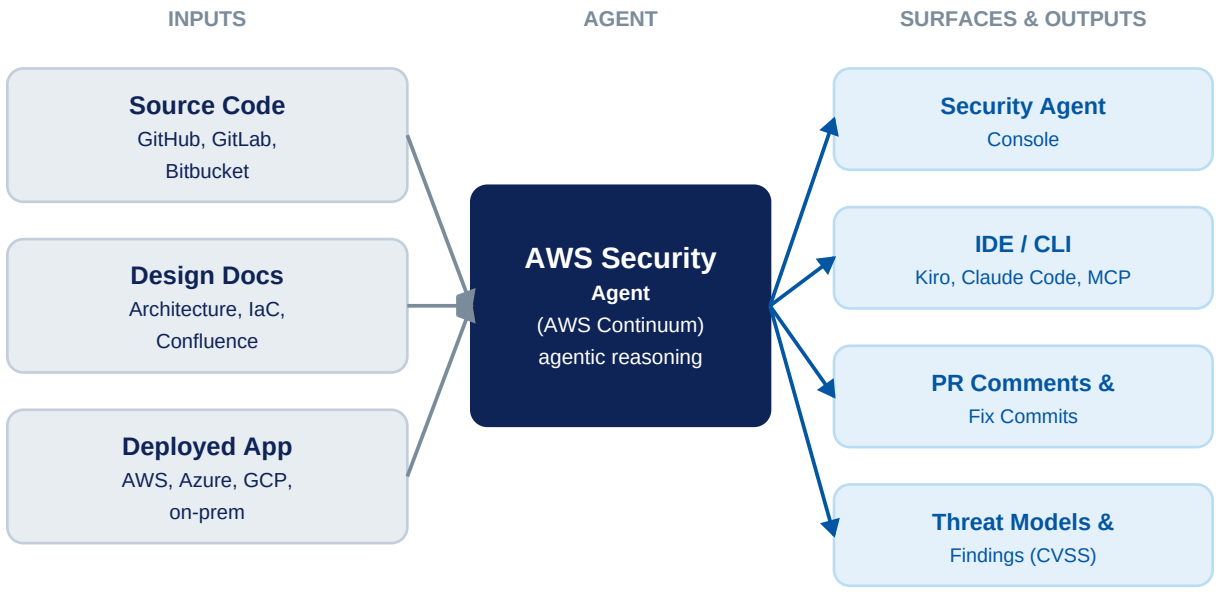

Figure 1. AWS Security Agent architecture — inputs, the agent core, and where results surface.

2. Problems It Helps Solve

Penetration testing doesn't scale

Manual penetration testing is expensive and slow, so most organizations only test their most critical applications, and only periodically. That leaves the majority of an application portfolio unassessed between test cycles. AWS Security Agent turns penetration testing into an on-demand, 24/7 capability, so every application can be tested whenever needed rather than waiting for a scheduled engagement.

Threat modeling requires scarce expertise

Building a rigorous threat model traditionally requires a security architect to manually map data flows, trust boundaries, and attack vectors — a time-consuming process that is often skipped or done superficially under deadline pressure. The agent automates this analysis directly from design documents or code.

Traditional scanners miss systemic issues

Conventional static analysis tools match code against known vulnerability signatures. They are effective against known patterns but blind to vulnerabilities that emerge from how an application's components interact. Security Agent's code review instead reasons about architecture and data flow to surface issues that pattern-matching tools cannot.

Security work creates delivery bottlenecks

Security reviews are often a late, manual gate before release, slowing delivery. By embedding review, threat modeling, and validated remediation directly into pull requests and IDEs, Security Agent lets teams catch and fix issues inline, without a context switch to a separate security tool or team hand-off.

Findings without proof of exploitability create noise

Security teams are often flooded with findings that are theoretical rather than exploitable, making prioritization difficult. Security Agent validates findings in simulated environments to demonstrate real exploitability, and provides CVSS and application-specific severity ratings so teams can focus on what actually matters.

Fragmented tools across the SDLC

Design review, threat modeling, code review, and penetration testing are traditionally separate tools, vendors, and workflows. AWS Security Agent unifies design-time, development-time, and deployment-time security into a single agentic offering with a consistent view of the application.

3. Core Capabilities in Detail

3.1 Design Review

Design review continuously validates architecture and design documentation against managed compliance packs — including AWS WAF, NIST CSF, and PCI DSS — or against an organization's own requirements imported from internal documents or Confluence. Every finding maps back to compliance posture, helping teams stay audit-ready as they build rather than scrambling before an audit.

3.2 Threat Modeling (Preview)

From design documents or a connected source code repository, the agent builds a contextual model of the application: its components, data flows, architecture, and trust boundaries. It then identifies potential threat actors and attack vectors, determines where weaknesses may exist, and prioritizes threats across all six STRIDE categories (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) so teams know what to address first.

3.3 Code Review (Preview)

Code review supports both per-pull-request scanning and full-repository analysis, connecting to GitHub, GitLab, and Bitbucket (SaaS and self-hosted), with Confluence integration for documentation context. Rather than matching known vulnerability signatures, it reasons about the application's trust boundaries and data flows to surface systemic issues, then generates fix commits and remediation guidance directly in the developer's existing Git workflow. Findings are validated in simulated environments before being surfaced, reducing false positives.

3.4 On-Demand Penetration Testing (Generally Available)

This is the most mature capability, having reached general availability on March 31, 2026. The agent ingests source code, architecture diagrams, and documentation to understand how an application was built, then acts like a human penetration tester: identifying potential vulnerabilities, attempting exploitation with targeted payloads and attack chains, and validating that findings are legitimate risks rather than theoretical ones. It connects individual vulnerabilities into higher-severity attack chains that traditional scanners typically miss, and supports AWS, Azure, GCP, other cloud providers, and on-premises environments, enabling consolidated testing across a full multicloud estate.

Reported Impact

Metric	Reported Result
Penetration testing timeline	Compressed from weeks to hours in preview feedback
Cost	A fraction of the cost of manual penetration testing
Coverage	24/7, on-demand — enables testing across the full application portfolio, not just top-tier apps

Metric	Reported Result
Findings quality	CVSS scores, app-specific severity ratings, reproduction steps, and remediation guidance

Reported figures come from AWS's own preview customer feedback and announcements; treat them as vendor-reported until validated against your own workloads.

4. How It Fits Into the SDLC

AWS positions Security Agent as covering the full software development lifecycle in a single, unified agentic offering:

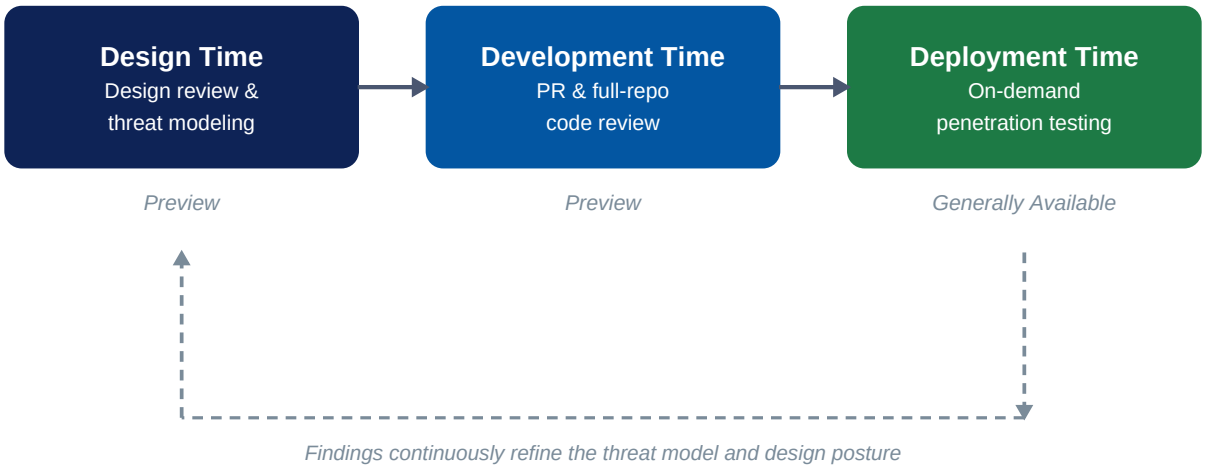

Figure 2. Security Agent capabilities mapped across the SDLC, with a continuous feedback loop from later stages back into design and threat modeling.

Phase	Security Agent Capability	Typical Trigger
Design time	Design review, threat modeling	New architecture doc, design change, repo connect
Development time	Code review (PR + full repo)	Pull request opened, scheduled repo scan
Deployment time	On-demand penetration testing	Pre-release validation, scheduled recurring test

Findings and context flow between phases: a threat model's trust-boundary analysis can inform which areas a penetration test should probe first, and code review findings can feed back into the threat model as the application evolves.

5. Step-by-Step Implementation Guide

The steps below reflect the general onboarding flow AWS documents for Security Agent as of mid-2026. Screens and exact menu names may shift as capabilities move from preview to general availability — check the AWS Security Agent User Guide for the current console flow.

Step 1 — Confirm regional availability and access

On-demand penetration testing is generally available in six commercial Regions: US East (N. Virginia), US West (Oregon), Europe (Ireland), Europe (Frankfurt), Asia Pacific (Sydney), and Asia Pacific (Tokyo). Design review, threat modeling, and code review are in preview and may have different availability. Check the AWS Capabilities by Region page for the current footprint before planning a rollout.

Step 2 — Open the Security Agent console

Sign in to the AWS Console and navigate to the Security Agent console. New customers can start with a 2-month free trial; review the Security Agent pricing page for usage-based cost details beyond the trial.

Step 3 — Connect your source of truth

Connect a source code repository (GitHub, GitLab, or Bitbucket — SaaS or self-hosted) and, optionally, Confluence for design documentation. This gives the agent the architecture and code context it needs to reason about your application rather than working from code alone.

Step 4 — Enable design review and set compliance packs

Turn on design review and select the compliance frameworks relevant to your organization — AWS WAF, NIST CSF, PCI DSS, or AWS best practices — or import your own internal requirements. Findings will map back to your compliance posture automatically.

Step 5 — Enable threat modeling

Choose "Enable threat model," then connect your source code repository if you haven't already. The agent will generate a STRIDE-based threat model mapping components, data flows, and trust boundaries, prioritized by risk.

Step 6 — Enable code review

Choose "Enable code review" (or update existing settings) in the console. Configure which repositories are monitored; the agent will scan pull requests and full repositories, returning fix commits and remediation guidance directly in your Git workflow. Security teams should configure which repos are monitored and set escalation paths for critical findings.

Step 7 — Run an on-demand penetration test

From the console (or CLI, once integrated — see Step 8), launch a penetration test against a target application. The agent will identify vulnerabilities, attempt exploitation, validate findings, and return a report with CVSS scores, severity ratings, reproduction steps, and remediation suggestions.

Step 8 — Integrate into your IDE and CI/CD workflow

For developer-facing workflows, install the Security Agent power for Kiro, or connect through the open MCP server (the Claude Code plugin is rolling out separately). This lets developers trigger threat models, code reviews, and remediation directly from the IDE or CLI — for example, asking an IDE agent to "Run a full security scan on this repo" or "help me remediate my findings" — without a context switch to a separate console.

Step 9 — Automate recurring tests and route findings

Use Amazon EventBridge and AWS Step Functions to schedule recurring penetration tests, with Amazon SNS notifications on completion. AWS publishes a sample CloudFormation template for exactly this pattern. Route critical findings to your existing ticketing or chat tools so security and engineering teams can triage quickly.

Step 10 — Establish a remediation and re-test loop

Treat findings as a workflow, not a one-time report: prioritize by severity, assign fixes, apply the agent's suggested remediations or generated fix commits, and re-run validation to confirm the issue is resolved before closing it out.

Prerequisites Checklist

- An AWS account with permissions to enable Security Agent / AWS Continuum services
- Access to your source code repository (GitHub, GitLab, or Bitbucket) with permission to install a scanning integration
- Identified target applications and environments (including any non-AWS clouds or on-prem systems to include)
- Agreement internally on which compliance frameworks matter (PCI DSS, NIST CSF, internal policy, etc.)
- A defined escalation path for critical findings before your first scan runs

6. Best Practices for Rollout

- **Start with one team or application** — Pilot on a single repository or application before expanding org-wide, so you can tune compliance packs and escalation paths first.
- **Pair automated findings with human review for critical issues** — Use the agent's validated, exploit-proven findings to prioritize, but keep a human-in-the-loop for production-impacting remediations.
- **Feed the agent good context** — Connect design docs and Confluence spaces where available; richer architecture context produces more accurate threat models and fewer false positives in code review.
- **Schedule recurring tests, don't rely on one-off runs** — Use EventBridge/Step Functions scheduling so penetration testing keeps pace with your release cadence rather than becoming another annual checkbox.
- **Bring security into the IDE** — Adopting the Kiro power, MCP integration, or Claude Code plugin keeps developers fixing issues where they already work, which meaningfully improves how quickly findings get resolved.

- **Track preview vs. GA status** — As of mid-2026, penetration testing is GA while design review, threat modeling, and code review are in preview; confirm current status before depending on any capability for compliance purposes.

7. Summary

AWS Security Agent represents a shift from point-in-time, tool-fragmented application security toward a single agentic system that understands an application's design, code, and behavior, and continuously acts on that understanding — reviewing designs, modeling threats, reviewing code, and validating exploitability through autonomous penetration testing. For security and engineering teams facing scaling pressure, its core value proposition is straightforward: security work that used to be periodic, expensive, and expert-gated becomes available on demand, at machine speed, embedded directly into existing development workflows.

Sources: AWS News Blog, AWS What's New announcements, and AWS Security Blog posts, published between December 2025 and June 2026. Capabilities, regional availability, and pricing evolve quickly for actively developed services — verify current details on the AWS Security Agent product page and technical documentation before making implementation decisions.