BLACK DUCK®

CHECKLIST

AI CODING ASSISTANTS: YOUR SEVEN-LAYER SECURITY CHECKLIST

The teams winning with AI are not the teams with the best prompts. They're the teams with the best guardrails. This seven-layer security checklist details the controls, processes, and governance needed to use AI coding assistants safely and effectively, from predevelopment planning through runtime operations.

LAYER 1: STANDARDS AND GROUNDING

Set up the foundation of your AI system by providing documentation of your coding and security standards.

- ☐ Document your organization's coding standards and provide them to the AI before generating any code.
- ☐ Give your organization's security standards and selected OWASP controls to the AI.
- ☐ Provide architecture decision records and security tradeoff documentation to the AI at the start of every session.
- ☐ Communicate regulatory and compliance requirements (PCI-DSS, HIPAA, SOC 2, etc.) to the AI at the start of every AI session that involves sensitive data flows.
- ☐ Use precise, security-explicit prompts that instruct the AI to optimize for safety, not just correctness.

LAYER 2: THREAT MODELING

Develop separate threat models for build time and runtime to limit insecure code generation and AI-specific risks.

- ☐ Create a build time threat model for AI-introduced vulnerabilities like insecure code generation, hallucinated packages, and prompt injection.
- ☐ Build a runtime threat model covering prompt injections, data leakage, hallucinations, and model supply chain risks.
- ☐ Model these agentic AI threat scenarios explicitly: autonomous commits, infrastructure changes, pipeline triggers, and permission grants without human authorization.

LAYER 3: AI-GENERATED TESTS AND SECURITY LOOPS

Create a security loop that requires the AI to run tests at every stage of development. Never treat AI-generated code as final.

- ☐ Require the AI to generate unit tests for all AI-generated code before it is considered complete.
- ☐ Include regression tests for every AI-generated bug fix to prevent the same vulnerability from re-entering the codebase.
- ☐ Require integration tests covering AI component boundaries, database interactions, external service calls, and trust boundary crossings.
- ☐ Separate bug reviews from security reviews. They require different prompts.
- ☐ Run tests at every stage of the AI coding cycle, not just before release.
- ☐ Verify that AI-generated tests exercise meaningful security properties, not just surface-level pass conditions.

LAYER 4: AI REVIEWS AI

Add a layer of scrutiny to every change before it reaches human reviewers by creating a separate AI pass specifically scoped to security.

- ☐ Deploy a dedicated AI security review for only changed code, not the entire codebase.
- ☐ Configure AI security reviews to run automatically on every pull request before human review begins.
- ☐ Make sure AI review output feeds into your developer workflow as actionable, annotated findings, not a separate security report that gets ignored.
- ☐ Track AI-flagged security findings over time and use the data to improve the grounding and standards documents provided to the system in Layer 1.

LAYER 5: SAST, SCA, AND DAST

Utilize the three main scanning technologies to ensure that remediation happens before any public exposure.

- ☐ Run SAST on every commit in a CI gate, blocking merges on any critical findings.
- ☐ Deploy IDE-based SAST scanning to catch insecure AI-generated code at the point of generation and before it enters the pipeline.
- ☐ Run SCA on every commit to detect hallucinated, nonexistent, and maliciously registered packages before they enter the build.
- ☐ Verify that every AI-suggested package exists in an approved registry before installation or before it is added to the dependency manifest.
- ☐ Run DAST against the running application to catch the gap between intent and behavior that even thorough code reviews can miss.
- ☐ Run SAST, SCA, and DAST on every commit if the pipeline can support it, and always before any major release.
- ☐ Ensure that AppSec tooling provides unified, cohesive findings across SAST, SCA, and DAST scans rather than siloed, disconnected outputs.

LAYER 6: HUMAN SECURITY REVIEWS

Have human developers and security experts use their valuable judgment to perform mandatory reviews.

- ☐ Require a security engineer or senior developer to review every AI-generated change for payment flows, authentication logic, and authorization boundaries.
- ☐ Require human review with explicit trust boundary analysis for all agentic AI workflows before granting them production access or real-world permissions.
- ☐ Train security reviewers to apply heightened scrutiny to AI-generated code by challenging assumptions, independently verifying outputs, and carefully assessing whether sensitive data is being handled appropriately.
- ☐ Maintain end-to-end visibility for development, DevOps, and AppSec teams across every stage of the SDLC, from IDE through runtime.

LAYER 7: RUNTIME DEFENSES

Defend runtime AI systems against attacks that don't occur in traditional deployments.

- ☐ Deploy runtime monitoring and anomaly detection specifically tuned to AI-generated application behaviors and agentic system actions.
- ☐ Check for prompt injections at inference time for any AI system that processes external, user-controlled, or third-party content.
- ☐ Monitor model supply chain integrity: Track model versions, providers, and API endpoints used by AI-enabled systems in production.
- ☐ Implement human-in-the-loop gates for all agentic AI actions with real-world consequences.
- ☐ Apply least-privilege principles to agentic AI systems: Grant the minimum permissions required and restrict access to production databases, secrets, and infrastructure by default.
- ☐ Maintain incident response runbooks that cover AI-related security events including prompt injection exploitation, model misbehavior, and hallucinated package compromise.

KEEPING IT ALL TOGETHER: CONTINUOUS GOVERNANCE

If these seven layers represent a cake, consider governance the frosting that holds everything together.

Maintain security by having a robust governance strategy that establishes AI usage policies, guidelines, and standards.

- ☐ Create and publish an organizational policy for approved AI coding assistants, usage boundaries, and data-handling requirements.
- ☐ Implement monitoring and detection of unauthorized AI tool usage within development environments.
- ☐ Establish a clear process for developers to request approval for new AI tools to reduce shadow IT.
- ☐ Formally assess and document your organization's current readiness to secure AI-generated code.
- ☐ Actively build a culture that treats AI output as a starting point requiring verification instead of a finished product ready for deployment.
- ☐ Recognize and reward developers who find and call out AI-generated security issues.
- ☐ Evolve DevSecOps practices in parallel with AI adoption to ensure that security is embedded at every stage of the AI-accelerated SDLC.

HOW BLACK DUCK CAN HELP

Black Duck solutions enable you to gain visibility across your SDLC, prioritize exploitable risk, and automate remediation with AI-driven, hybrid application security. This leads to faster development with lower risk and scalable security operations in the face of AI-driven vulnerability growth.

With Black Duck you can

- **Achieve complete visibility and eliminate oversights:** Identify all vulnerabilities across code, OSS, containers, and AI-generated components
- **Shift to risk-based prioritization:** Focus on exploitable risk first, not severity alone
- **Automate remediation with VulnOps:** Move from detection to machine-speed prioritization and resolution
- **Adopt agentic, AI-driven security operations:** Continuously identify, validate, prioritize, and remediate vulnerabilities

About Black Duck

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.