From IDE to Run Time: Securing Every AI Touchpoint

Praneetha Goli
Senior AI Engineer, Capital One

From IDE to Runtime

"AI-native engineering requires AI-native trust architecture."

80%

of new devs adopt AI tools in week one

(GitHub Octoverse, 2025)

45%

of AI code fails security tests

(Veracode, 2025)

81%

of security teams: zero AI visibility

(Cycode, 2025)

Why This Matters Now: The AI Coding Explosion

AI is Becoming a Primary Author and the Attack Surface Now Spans Every Layer From IDE to Runtime

AI CODING ADOPTION IS UNIVERSAL

GitHub Copilot
Embedded across enterprise IDEs

Cursor
Agentic editing and multi-file generation

Windsurf
Autonomous workflow agents

Claude Code / Codex CLI
Terminal and repo-level autonomy

FROM ASSISTANT TO AUTHOR

2022–23: Autocomplete
AI suggests single lines; human writes everything

2024: Assistant
AI drafts functions; human reviews each block

2025: Co-Author
AI generates multi-file changes; human guides

2026+: Primary Contributor
AI agents author and commit autonomously

45%

of AI-generated code fails security tests

Veracode 2025 GenAI Report (100+ LLMs tested)

Why Security Leaders Should Care

AI Accelerates Software Delivery and Accelerates the Creation and Propagation of Security Debt

VELOCITY

2–4×

Faster development

AI agents commit in seconds

Release cycles compress

Human review cannot keep pace

How fast can you detect what you didn't have time to review?

RISK

2.74×

More vulns in AI code vs. human

More code than humans can review

Hallucinated dependencies

Prompt injection targets agents

Are your controls built for code, or for the process?

GOVERNANCE

81%

Zero AI visibility

Agents gain privileged access

Shadow AI bypasses tooling

Agent identity absent from IAM

Can you prove accountability for AI-generated changes?

Threat Landscape: A New Class of Attack

These Are Not Bugs in Code. They Are Attacks on Reasoning, Tool Use, and Autonomous Action.

CRITICAL	**Prompt Injection** Malicious instructions in repo content, PRs, or URLs the agent executes as commands. OWASP LLM Top 10 #1 · 85% success rate against current defences (arXiv, 2025)
CRITICAL	**Tool Abuse** Manipulated tool descriptions cause agents to perform unintended privileged actions
HIGH	**Data Poisoning** Corrupted training data, RAG sources, or memory cause consistently insecure outputs
HIGH	**Slopsquatting** AI hallucinates a package name; attacker pre-registers it; developer ships malware. Documented by Socket.dev, 2024 · Traditional SCA misses these entirely
HIGH	**Agent Misbehavior** Autonomous agents take unintended actions through reasoning errors or excessive agency

MCP and the AI Supply Chain

MCP Introduces a Growing Ecosystem that Requires the Same Supply-Chain Security Discipline as Software Dependencies

THE PATTERN REPEATS

SOFTWARE SUPPLY CHAIN	AI SUPPLY CHAIN
npm / PyPI packages	MCP servers
Transitive dependencies	Third-party agent extensions
Typosquatting	Tool description poisoning
Dependency confusion	Unverified tool registries

Same attack patterns. New ecosystem. Far less mature security tooling. 57% of orgs now have AI agents in production (LangChain, 2026) — Shadow AI bypasses every control layer.

AI SUPPLY-CHAIN COMPONENTS

MCP Servers
Tool-providing servers agents connect to — growing rapidly, minimal vetting

Tool Registries
Marketplaces of agent capabilities with no standardised security review

Agent Extensions
Third-party plugins that inherit agent context and permissions

Model Artifacts
Fine-tunes, LoRA adapters, open-weight models with unverified provenance

Apply supply-chain discipline: approved registries · provenance review · version pinning · least-privilege scoping

Threat Surface Expansion: Every Hop is a Vector

Four of five layers sit outside what traditional AppSec was built to protect.
The surface expanded; the controls did not.

A single compromised hop propagates downstream: one prompt injection at the IDE can reach cloud infrastructure

WHERE TRADITIONAL APPSEC STOPS · WHERE AI-NATIVE CONTROLS MUST BEGIN

Code-level	Reasoning-level	Tool-level	Behaviour-level	Identity-level
SAST / SCA cover this layer well ✓ covered	No traditional control exists ✗ gap	Supply-chain controls needed ✗ gap	Runtime enforcement required ✗ gap	NHI & scoped access required ✗ gap

AI Security Touchpoints: Securing the Journey

Security Must Be Present at Every Touchpoint of the Developer Journey, Not Bolted On at the End

IDE	CI/CD	AGENT	RUNTIME
Inline AI-aware SAST + safe-fix at generation time	Merge gates: block on SAST / secret / SCA failure	Tool authorization — allowlisted tools only	Deterministic policy enforcement + RASP
Governance files: AGENTS.md / CLAUDE.md or equivalent	Dependency allowlists — slopsquatting defence	Action logging — every call c aptured for audit	Behavioural monitoring and drift detection
Scoped agent identity (NHI) — zero prod-level trust	SLSA attestation: source-to-artifact traceability	Approval gates for irreversible operations	Reachability analysis — exploitable vulns first
Human-in-loop for shell / IAM / infra changes	Policy-as-code: approved models, tools, merge rules	Agent BOM: extends the SBOM — tracks model version, prompt lineage, every tool invoked	IR playbooks for agent failures + action replay

Non-Human Identities: Identity ≠ Trust

Authentication Alone Does Not Establish Agent Trust. An Authenticated Agent Can Still Be Hijacked By Prompt Injection.

DIMENSION	TRADITIONAL IAM	AI-AGENT IAM (required now)
Identity type	Human user	Non-Human Identity (NHI)
Access model	RBAC + MFA	Tool permissions + scoped IAM roles
Credentials	Password / SSO token	Short-lived tokens / SPIFFE workload identity
Verification	Auth at login	Human approval gate + attestation
Threat model	Account takeover	Prompt injection → privilege escalation

Runtime Security: The Final Validation Layer

Alignment and System Prompts are Not Security Boundaries. Runtime Enforcement is Deterministic and Final.

Runtime Monitoring

Observe agent actions, tool calls, and data access in production — continuously, not periodically

Behaviour Analysis

Detect abnormal tool usage, credential access, and unusual action sequences in real time

Drift Detection

Monitor whether agent behaviour shifts over time — model updates, context drift, gradual change

Continuous Evaluation

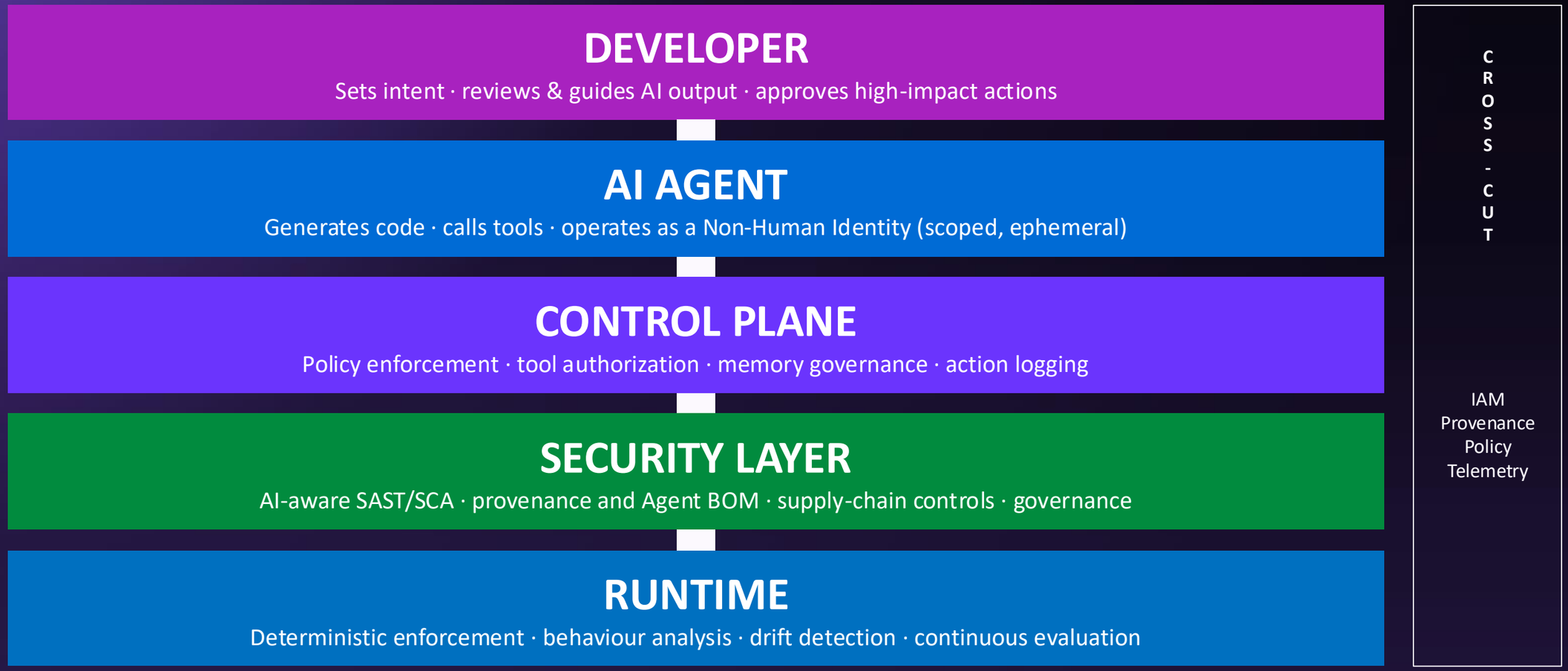
Treat evaluation pipelines as security infrastructure — red-team and regression-test agents continuously

The shift:

"Is this code vulnerable?" → "Is this behaviour exploitable, and is the agent still acting as intended?"

Future State: The Secure AI Development Architecture

Secure the Code. Secure the Agent. Secure the Runtime. Defense in Depth, from IDE to Runtime.



3 Things You Can Do This Week

1

Inventory: Know What AI is Running in Your Environment

Scan SSO logs, network traffic, and developer machines to map every AI tool, MCP server, and API key in use. Add AGENTS.md governance files to your highest-risk repos. You cannot secure what you cannot see.

2

Red-Team: Find the Gaps Before Attackers Do

Run prompt injection exercises against your top 3 agentic workflows this week. Target MCP server descriptions, agent context inputs, CI prompts, and README files. Block autonomous merges on auth, payment, and IAM files immediately.

3

Audit: Track Your Model Supply Chain Like You Track Code

For every model in production, answer two questions: where did it come from, and has it been modified? If you cannot answer both, you have an untracked supply chain artefact — treat it like an unpatched Log4j.

The Path Forward

The future of AppSec is not securing code written by humans.

It is governing autonomous systems that create software.

At every touchpoint, from IDE to runtime.

The organizations that treat AI agents as just another developer will be the ones that get breached.

The ones that secure every touchpoint from IDE to runtime will lead.

References

AI ADOPTION & DEVELOPMENT VELOCITY

[1] GitHub Octoverse 2025 Report — "80% of new developers use Copilot within their first week"
github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/

[2] LangChain — State of Agent Engineering 2026 — "57% of respondents have agents in production"
langchain.com/state-of-agent-engineering

AI CODE SECURITY

[3] Veracode — 2025 GenAI Code Security Report — "45% of AI code fails security tests; 2.74× more vulns than human code"
veracode.com/resources/analyst-reports/2025-genai-code-security-report/

[4] Cycode — State of Product Security for the AI Era 2026 — "81% of security teams lack AI visibility"
cycode.com/press/report-shadow-ai-crisis-looms-as-100-of-companies-have-ai-generated-code-but-81-of-security-teams-lack-visibility/

THREAT RESEARCH

[5] arXiv:2601.17548 — "Prompt Injection Attacks on Agentic Coding Assistants" (Jan 2026) — 42 attack techniques, 85%+ success rate
arxiv.org/abs/2601.17548

[6] Socket.dev — "The Rise of Slopsquatting: How AI Hallucinations Are Fueling a New Class of Supply Chain Attacks" (April 2025)
socket.dev/blog/slopsquatting-how-ai-hallucinations-are-fueling-a-new-class-of-supply-chain-attacks

STANDARDS & FRAMEWORKS

[7] OWASP Top 10 for LLM Applications 2025 — llmtop10.com

[8] SLSA Framework (Supply-chain Levels for Software Artifacts) — slsa.dev

[9] SPIFFE — Secure Production Identity Framework for Everyone — spiffe.io

[10] Anthropic — Model Context Protocol (MCP) Specification — modelcontextprotocol.io

Agentic AppSec Unleashed '26

by Checkmarx