encore

The Compliance Guide to AI-Written Code

What engineering leaders need in place under SOC 2, ISO 27001, and the EU AI Act.

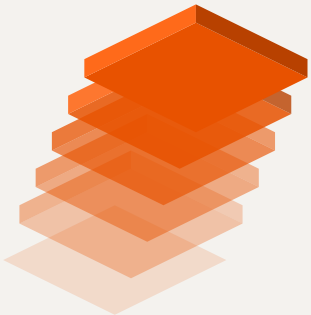


Table of Contents

Executive Summary

Three numbers describe AI-assisted development in 2026. At Google, 75% of new code is now AI-generated and approved by engineers, up from a quarter in late 2024.[1] Across the industry, 34% of organizations say more than 60% of their code is AI-generated, and only 18% have policies governing the use of the tools that write it.[2]

The distance between those numbers is governance debt, and it is now producing incidents, leaked credentials, and compliance exposure on a measurable scale. The instinctive fix, reviewing everything harder, collides with review pipelines that are already the bottleneck.

This guide is an operating manual for closing that gap: what the evidence says AI-generated code actually does in production, where secrets leak, how much autonomy to grant agents by change class, what a workable policy contains, and how to argue the compliance case under SOC 2, ISO 27001, and the EU AI Act, which becomes fully applicable on August 2, 2026.

1 Sundar Pichai, Google Cloud Next 2026 ("75% of all new code at Google is now AI-generated and approved by engineers"); Alphabet Q3 2024 earnings ("more than a quarter").

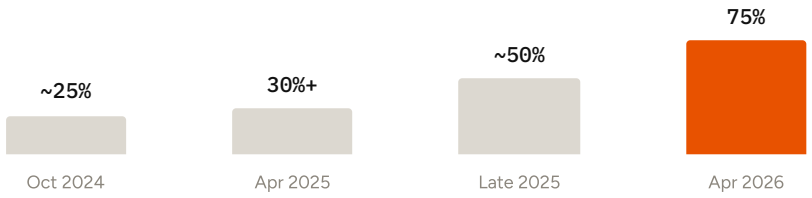
2 Checkmarx, "Future of Application Security in the Era of AI", Aug 2025, 1,500+ CISOs, AppSec managers and developers.

1 The Governance Gap

The clearest public record of how fast AI took over code production comes from Google's own statements: a quarter of new code in October 2024, well over 30% by April 2025, about half by late 2025 (written by coding agents and reviewed by engineers), and 75% by April 2026.[1] Microsoft's Satya Nadella put his company's share at 20 to 30% back in April 2025.[2]

FIGURE 01 — SHARE OF NEW CODE AT GOOGLE THAT IS AI-GENERATED

GOOGLE'S PUBLIC STATEMENTS



All figures describe code that is AI-generated and reviewed or approved by engineers. Sources: Alphabet earnings calls Q3 2024, Q1 2025, Q4 2025; Google Cloud Next 2026.

The pattern extends well beyond Big Tech. In Checkmarx's survey of 1,500+ security and development leaders, 34% of organizations report that more than 60% of their code is AI-generated.[3] For a large share of the industry, AI generation is already the default way code gets written.

Policy coverage has not kept pace

FIGURE 02 — HOW FAR POLICY TRAILS ADOPTION

THREE SURVEYS, 2025

18%
HAVE POLICIES GOVERNING THE USE OF AI CODING ASSISTANTS[3]

28%
OF ORGANIZATIONS WORLDWIDE HAVE A FORMAL, COMPREHENSIVE AI POLICY[4]

26%
HAVE COMPREHENSIVE AI SECURITY GOVERNANCE POLICIES[5]

Sources: Checkmarx (n=1,500+), ISACA (n=3,029), Cloud Security Alliance with Google Cloud (n=300).

Three independent surveys converge on the same finding: adoption is running roughly three times ahead of policy, and every percentage point of AI-generated code that ships without governance adds exposure. The rest of this guide covers how to close that gap with controls that hold at agent output volume, rather than by slowing adoption down.

1 Alphabet Q3 2024 earnings (Pichai: "more than a quarter"); Alphabet Q1 2025 earnings call ("well over 30%"); Alphabet Q4 2025 earnings call (CFO: "about 50%... written by agents... reviewed by our own engineers"); Google Cloud Next 2026 (Pichai: "75% of all new code at Google is now AI-generated and approved by engineers").

2 Satya Nadella, LlamaCon fireside chat, Apr 29, 2025 ("maybe 20%, 30% of the code that is inside of our repos today").

3 Checkmarx, "Future of Application Security in the Era of AI", Aug 2025, n=1,500+.

4 ISACA, 2025 AI Pulse Poll, 3,029 digital trust professionals worldwide.

5 Cloud Security Alliance with Google Cloud, "The State of AI Security and Governance", Dec 2025, n=300.

2 What AI-Generated Code Does in Production

A governance policy is only as good as its model of the risk. The research base from 2023 to 2026 is unusually consistent about what AI-generated code does when nobody is checking.

It carries more security defects

Veracode tested 100+ models against 80 curated coding tasks: 45% of AI-generated code introduced security vulnerabilities, with an 86% failure rate on cross-site scripting. Newer and larger models wrote more capable code without writing more secure code, and Veracode's Spring 2026 update, now covering 150+ models, puts it plainly: two years of model releases moved the security pass rate from roughly 55% to roughly 55%.[6] CodeRabbit's analysis of 470 real open-source pull requests found AI-co-authored PRs carried roughly 1.7x more issues than human-only ones, and were 2.74x more likely to introduce XSS.[7] Sonar's lab study makes the sharpest version of the point: when one leading model improved its benchmark pass rate by 6.3%, its high-severity bug rate rose 93%.[8]

It inflates developer confidence

THE OVERCONFIDENCE PATTERN

In Stanford's controlled study, developers with an AI assistant wrote significantly less secure code overall, producing insecure solutions more often on four of five tasks, and were more likely to believe their code was secure.[9] In METR's randomized trial, experienced open-source developers took 19% longer when using AI tools, while believing the AI had sped them up by 20%.[10] The defects arrive together with miscalibrated trust, in both security and speed.

Practitioners sense this: DORA's 2025 research found 30% of developers report little to no trust in AI-generated code, even as 90% use AI at work.[11] The result is an uneasy equilibrium where output volume grows, formal controls lag, and individual judgment carries the difference.

Staging does not catch it

In a 2026 survey of 200 senior SRE and DevOps leaders at large enterprises, 43% of AI-generated code required manual debugging in production after passing QA and staging tests, and not one respondent said they were very confident that AI code behaves correctly once deployed.[12] Pre-production gates were designed for code written by someone with a mental model of the system. AI code satisfies the tests it can see and fails on the intent it cannot.

6 Veracode, "2025 GenAI Code Security Report" (80 tasks, 100+ LLMs) and "Spring 2026 GenAI Code Security Update" (150+ models). veracode.com

7 CodeRabbit, "State of AI vs. Human Code Generation", Dec 2025, 470 open-source PRs.

8 Sonar, "The Coding Personalities of Leading LLMs", Aug 2025, over 4,400 Java assignments, 5 LLMs.

9 Perry et al., "Do Users Write More Insecure Code with AI Assistants?", ACM CCS 2023, n=47.

10 METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity", Jul 2025 (arXiv:2507.09089).

11 Google Cloud DORA, "State of AI-assisted Software Development", 2025.

12 Lightrun, "The State of AI-Powered Engineering 2026", Global Surveyz, n=200 senior SRE/DevOps leaders, US/UK/EU enterprises 1,500+ employees.

When it reaches production systems

CASE STUDY: AMAZON, MARCH 2026

Amazon's retail site suffered a series of outages in one week. By Amazon's own account, one incident involved an engineer following inaccurate advice that an AI tool inferred from an outdated internal wiki. Internal documents reported by the Financial Times and CNBC described a "trend of incidents" tied to "Gen-AI assisted changes"; Amazon disputes parts of that reporting and says it has updated internal guidance.[13]

What failed here was the advice path rather than a generated diff: an agent read a stale internal document, a human trusted the output, and a production system inherited the error. Governance that only reviews diffs would not have caught it. The scope of a working policy is everything an AI system touches on the way to a change: sources, suggestions, code, and configuration.

Why "review everything harder" fails

The instinctive response to all of the above is to add human review, but the capacity for that does not exist. Telemetry from 8.1 million pull requests shows AI-generated PRs already wait 4.6x longer for first review, and their acceptance rate is 32.7% versus 84.4% for manual PRs.[14] Review capacity is the scarcest resource in an AI-accelerated organization. A governance model that spends it uniformly, on every change regardless of risk, guarantees both a bottleneck and reviewer fatigue on exactly the changes that deserve attention.

FIGURE 03 — WHAT REVIEW CAPACITY ALREADY LOOKS LIKE

8.1M PULL REQUESTS

4.6x	32.7%	84.4%
LONGER WAIT FOR FIRST REVIEW, AI-GENERATED PRS	ACCEPTANCE RATE, AI-GENERATED PRS	ACCEPTANCE RATE, MANUAL PRS

Source: LinearB, "2026 Software Engineering Benchmarks Report", 8.1M+ pull requests across 4,800+ organizations.

The alternative is to spend review where risk lives and remove entire risk categories structurally. Chapters 3 through 5 cover that alternative: cutting off secrets exposure, setting autonomy by change class, and writing the policy that encodes both.

> "A high-quality platform amplifies the effects of AI adoption on organizational performance. The positive impact of AI on organizational performance is strong when platform quality is high."
>
> DORA Research
>
> 2025 STATE OF AI-ASSISTED SOFTWARE DEVELOPMENT

13 Amazon, aboutamazon.com, Mar 2026; Fortune, Mar 12, 2026; Financial Times and CNBC reporting; Wharton Accountable AI Lab, Apr 2026.

14 LinearB, "2026 Software Engineering Benchmarks Report", 8.1M+ pull requests, 4,800+ organizations.

3 The Secrets Problem

Of everything AI tools change about security posture, credentials exposure is the most measurable, and the numbers are moving the wrong way. GitGuardian's 2026 telemetry across public GitHub is the reference dataset.[15]

FIGURE 04 — CREDENTIALS EXPOSURE IN THE AI ERA

PUBLIC GITHUB, 2025–2026

2x

SECRET-LEAK RATE OF AI-ASSISTED COMMITS VS. ALL PUBLIC COMMITS (3.2% VS 1.5%)

28.65M

NEW HARDCODED SECRETS ON PUBLIC GITHUB IN 2025, +34% YOY, THE LARGEST JUMP ON RECORD

24,008

UNIQUE SECRETS EXPOSED IN MCP-RELATED CONFIGURATION FILES, 2,117 OF THEM VALID

Source: GitGuardian, "The State of Secrets Sprawl 2026", telemetry across public GitHub.

AI-service credentials themselves grew 81% year over year as teams wired up model providers and agent frameworks, and eight of the ten fastest-growing leak detectors were tied to AI services.[15] The mechanism is mundane: AI tools read whatever the repository and their configuration give them, and they reproduce patterns they see. If credentials are reachable, they end up in generated code, in commits, and in context sent to third parties.

The exposure map

EXPOSURE PATH	WHAT LEAKS	WHY AI MAKES IT WORSE
Repository files	.env files, IaC state, connection strings	Every AI tool reads the repo as context. Reachable credentials become training material for the next suggestion
Context windows	Whatever is pasted or auto-included in prompts	Production data and keys leave your perimeter to a third-party service, often unlogged
Agent configuration	MCP and tool configs holding tokens	24,008 secrets found in MCP-related config files on public GitHub[15]
Internal docs and wikis	Stale runbooks, credentials in docs	Agents treat internal documents as ground truth. The Amazon incident began with an outdated wiki[13]

Across all four rows, scanning and training reduce how often leaks happen, and only removing credentials from the places AI can read eliminates the class. Chapter 7 covers how to do that removal.

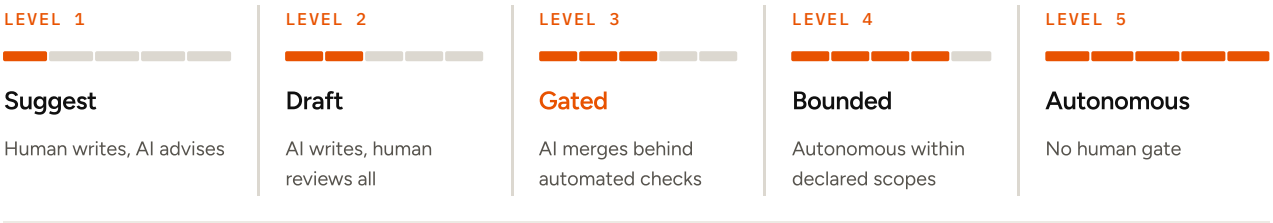
[15] GitGuardian, "The State of Secrets Sprawl 2026". gitguardian.com

4 How Much Autonomy Should Agents Have?

The question now facing engineering organizations is what an agent may do without a human in the loop, and the answer should depend on what is being changed rather than on which team is asking. Five levels cover the practical range:

FIGURE 05 — FIVE LEVELS OF AGENT AUTONOMY

SUGGEST → AUTONOMOUS



Autonomy rises left to right. The ceiling that applies depends on the change class, not on the team asking. Illustrative, not to scale.

Most organizations today run everything at level 2 and exhaust their reviewers, or let individual teams drift to level 4 without deciding to; both are policy failures. The workable pattern is an autonomy ceiling per change class:

CHANGE CLASS	EXAMPLES	AUTONOMY CEILING
Docs, tests, tooling	Documentation, test additions, internal scripts	Level 3-4: merge behind automated checks
Reversible app code	Feature code behind flags, UI, isolated services	Level 3: automated gates plus async human review
Sensitive app code	AuthN/authZ, payments, PII handling, data migrations	Level 2: mandatory human review before merge
Infrastructure and config	IAM, networking, provisioning, schemas	Level 1-2, or remove the surface: derive infrastructure from code (ch. 7)
Secrets and credentials	Keys, grants, rotation	Not in agent scope at all. Platform-managed

The cautionary tale for skipping this exercise: 33% of infrastructure teams say they would apply AI-generated HCL directly to production without any review.[16] That is level 5 autonomy on the highest-blast-radius change class, adopted by default rather than by decision. DORA's finding frames the payoff of doing it deliberately: AI amplifies the system around it, and the organizations that profit are the ones whose control systems were designed for it.[11]

16 Spacelift, "The Infrastructure Automation Report 2026: The AI Readiness Gap", n=406, fielded by Panterra Group.

5 Writing the Policy

The 18% of organizations with a policy did not get there by writing long documents. A workable AI development policy fits in a few pages and answers seven questions; everything else is implementation detail. DORA's 2025 guidance for leaders starts in the same place, with "clarify and socialize your AI policies" as its first recommendation.[11]

COMPONENT	THE QUESTION IT ANSWERS
Scope and tool allowlist	Which AI tools and agents are approved, and who approves new ones?
Data boundaries	What may AI tools read? What must never enter a context window or tool configuration?
Change classes and autonomy ceilings	What may an agent do without a human, per class of change? (Chapter 4's table.)
Review requirements	Which changes need human review, at what depth, and by whom? Where is review replaced by automated gates?
Attribution and audit	How is AI involvement recorded per change, and can you reconstruct who and what produced any line in production?
Incident handling	When a change causes an incident, how is AI involvement identified, and what feeds back into the ceilings above?
Exceptions and ownership	Who owns the policy, how are exceptions granted, and when is it revisited?

Starter rules that hold up in practice

- → Every AI-assisted change is labeled as such in commit or PR metadata. Attribution is the precondition for every other control on this list.
- → AI tools never hold production credentials, enforced through architecture rather than through instructions to the tool (chapter 7).
- → Autonomy ceilings attach to change classes, not to teams or tools. A new agent inherits the existing ceilings.
- → Treat review capacity as a budget. If agent output exceeds what humans can meaningfully review, lower the autonomy ceiling or raise the automated gates rather than letting review quality silently degrade.
- → Every agent action has a named human owner, so accountability for any change traces to a person.
- → Incident postmortems record whether AI was involved in the change and in the diagnosis. The ceilings move based on that record.

THE POLICY IN ONE SENTENCE

If a change cannot be attributed, gated, and rolled back, it does not ship. The rule applies to humans and agents alike, and with agents the delivery system itself can enforce it rather than relying on convention.

6 SOC 2, ISO 27001, and the EU AI Act

Engineering leaders keep asking what their auditors expect for AI-generated code, and as of mid-2026 there is no authoritative standard to point them to. Schellman, one of the largest SOC 2 audit firms, wrote in April 2026 that "we have nothing authoritative to cite from the AICPA at this time." NIST's secure-development guidance for AI (SP 800-218A) governs building AI systems, not code written by AI, and its agent-focused control overlays are still drafts.[17] In the absence of a standard, auditors will judge how convincingly your existing controls extend to AI-assisted changes.

The regulatory clock

FIGURE 06 — THE REGULATORY CLOCK — EU AI ACT

AUG 2, 2025	EU AI Act obligations for general-purpose AI models apply. Governance rules for the model providers your tools are built on.
AUG 2, 2026	The AI Act becomes fully applicable, including its transparency obligations. Enforcement powers are live.
2027-2028	High-risk system obligations phase in (pushed to Dec 2027 and Aug 2028 for certain categories under the simplification agreement).

Source: Regulation (EU) 2024/1689, Art. 50 and Art. 113, and the European Commission implementation timeline.

Most engineering organizations are deployers of AI systems rather than providers, which keeps their direct obligations modest. For coding tools specifically, the Commission's draft Article 50 guidelines (May 2026) exempt generated source code from the AI-content marking obligation, while natural-language output from the same tools, such as documentation, remains in scope. It is draft guidance, but it is the first regulatory line drawn through AI-generated code.[18] The near-term exposure is indirect: data protection law already applies when AI tools read production credentials and regulated data, and customers' security questionnaires are already asking AI-governance questions that the 18% can answer and the 82% cannot.

What your auditor will ask

- → Can you identify which production changes were AI-generated or AI-assisted?
- → Who approved each change, and does the approval depth match your stated policy?
- → What could the AI read at the time it produced the change? Could it access credentials or regulated data?
- → How do you detect, attribute, and roll back a defective AI-assisted change?
- → Does your incident process record AI involvement, and is there evidence the findings feed back into policy?

Every question maps to a control principle that predates AI: change management, least privilege, auditability, and incident response. The organizations that answer these questions well are the ones whose architecture makes the answers true by default, which is the subject of the next chapter.

17 Schellman, "What Does the AICPA Require of Artificial Intelligence?", updated Apr 2026; NIST SP 800-218A, Jul 2024; NIST Control Overlays for Securing AI Systems (COSAiS), drafts 2025-2026.

18 Regulation (EU) 2024/1689, Art. 50 and Art. 113; European Commission, draft Guidelines on Article 50 transparency obligations, May 8, 2026.

7 Governance by Construction

Every control in this guide comes in two versions: one that depends on human attention, and one the architecture enforces. At the output volume agents produce, only the second kind holds. Policies, training, and scanning reduce the frequency of failures, while removing the failure surface eliminates the class.

FIGURE 07 — THE SAME CONTROL, TWO WAYS TO ENFORCE IT

REVIEW VS CONSTRUCTION

CONTROL	GOVERNED BY REVIEW	GOVERNED BY CONSTRUCTION
Secrets exposure	Scanning, training, pre-commit hooks	Credentials live in the platform, outside the repo, leaving nothing for AI tools to read or leak
Infrastructure changes	Humans review AI-written IaC	Infrastructure derived from typed declarations in application code. No IaC surface for agents to touch
Unsafe patterns	Reviewer knowledge, checklists	Typed resources and least-privilege defaults. Unsafe configurations fail at compile time
Environment drift	Release checklists, staging sign-off	Local, preview, and production derived from the same source, identical by construction
Audit trail	Commit discipline, manual records	Every resource and deployment traceable to the code that declared it, automatically

The left column depends on human attention holding at agent output volume. The right column removes the failure surface instead.

This is the model Encore implements: infrastructure semantics live in the application code as typed declarations, the platform provisions and operates the resources, and credentials and infrastructure state never enter the repository. AI tools, including agents, work in the one place they measurably perform well, application code, inside guardrails that do not depend on human attention. The compliance conversation changes accordingly: "the AI never had access to production credentials" becomes an argument an auditor can verify by construction rather than a promise backed by training records.

> "In conventional IaC workflows, AI systems require access to environment variables and infrastructure configuration to diagnose issues, which creates compliance challenges under data protection law. With Encore's code-first approach, infrastructure is managed independently, so the AI can assist effectively without accessing sensitive configuration or regulated data."
>
> **Leonardo Marciano**
>
> CTO & FOUNDER · AI AND CYBERSECURITY

8 Getting Started

Answer each question yes or no. Where the noes cluster matters more than how many there are.

QUESTION	YES / NO
Do you know what share of your merged code is AI-assisted?	Y / N
Is AI involvement labeled on every commit or pull request?	Y / N
Is there an approved-tool list, with an owner for adding to it?	Y / N
Are data boundaries defined: what AI tools may and may not read?	Y / N
Are production credentials inaccessible to AI tools by construction?	Y / N
Do autonomy ceilings exist per change class, in writing?	Y / N
Is review capacity measured against AI output volume?	Y / N
Can you roll back any AI-merged change, and attribute it?	Y / N
Do incident postmortems record AI involvement?	Y / N
Does one named person own the AI development policy?	Y / N

HOW TO READ YOUR ANSWERS

Close the attribution and data-boundary noes first, because every other control depends on knowing what AI touched and what it could read. Noes on autonomy ceilings and review capacity mean the policy exists on paper but not in the pipeline, and a no on credentials is the largest single exposure.

FIGURE 08 — THREE MOVES, IN ORDER

MEASURE → REMOVE

STEP 01

Measure

Instrument AI attribution first. Two weeks of labeled commits shows your real exposure.

STEP 02

Set ceilings

Adopt the chapter 4 ceilings and publish them. That ends the silent drift toward unreviewed autonomy.

STEP 03

Remove the biggest surface

Move credentials and infrastructure state out of AI reach by platform choice. It is the only risk class you can eliminate outright.

Attribution comes first because every later control depends on it. Illustrative, not to scale.

THE AUGUST 2026 DEADLINE

The EU AI Act becomes fully applicable on August 2, 2026, and customer security questionnaires already ask AI-governance questions. The gap between the 18% with a policy and everyone else is now a sales and audit differentiator.

About Encore

Encore is a development platform for the AI era, built for a world where AI multiplies software delivery 10-100x rather than making each developer slightly faster.

In a traditional Terraform-based workflow, much of the infrastructure code cannot be meaningfully validated until it is applied to a real cloud environment, so mistakes surface late, in staging or production, where they are slower, riskier, and more expensive to fix. AI does not solve this by producing Terraform at roughly human quality: if the error rate stays the same while change volume grows 10-100x, the issues reaching the deployment loop grow 10-100x with it. At that scale, generated infrastructure would need to be effectively perfect rather than merely human-level.

Encore changes the model instead. Developers and AI agents declare infrastructure requirements, such as databases, queues, and scheduled jobs, directly in ordinary TypeScript or Go, and Encore derives the configuration, permissions, environments, and deployment setup from those declarations. The same application model runs locally, in isolated preview environments, and in production, so changes are validated with real infrastructure before they reach the production loop. Credentials, cloud configuration, and infrastructure state live in the platform, outside the repository and outside the reach of AI tools.

WHAT AI TOOLS SEE	WHAT THE PLATFORM ENFORCES	WHAT THE AUDIT TRAIL SHOWS
• Business logic and typed resource declarations	• Type-safe infrastructure semantics at compile time	• Every resource traceable to the code that declared it
• No credentials, no cloud config, no state files	• Least-privilege defaults on every resource	• Every deployment attributable and reversible
• Full application context for generating deployable code	• Identical local, preview, and production environments	• Architecture diagrams generated from the code

Where this fits your governance program

SITUATION	GOVERNANCE CHALLENGE	HOW ENCORE HELPS
Scaling agent use	Agent output growing faster than review capacity	Platform gates replace manual review for infrastructure concerns; agents work in application code only
Compliance-driven teams	Proving control over AI-assisted changes under SOC 2 / ISO 27001	Credentials and infra state outside AI reach by construction; automatic per-resource audit trail
Post-incident hardening	An AI-assisted change caused an outage	Typed guardrails and identical environments remove the classes of error reviews keep missing

> "Encore lets us scale both our product and infrastructure footprint, without additional hiring. The ROI we've seen is outstanding, easily 10x."
>
> **Daniel Stocks, CTO**
>
> CARLA · THE CASE STUDY REPORTS ANNUAL SAVINGS OF OVER $200,000 ON HEADCOUNT COSTS

Start building with Encore

Give your developers and agents infrastructure with the guardrails built in.

Get started free

Book a demo