tines

ESSENTIAL GUIDE

Taming wild code:

The IT leader's guide to AI code sprawl

Contents

Foreword

AI hasn't just made building faster – it's made everyone a builder.

Across every department, employees are shipping apps, agents and automations using AI tools, often unaware these builds need to be governed.

As a result, AI code sprawl is taking root, increasing risk, compromising compliance, and wasting resources. The usual responses aren't working: not because they're poorly executed, but because they were never built for a problem at this scale or speed.

Drawing on insights from experienced CIOs, this guide gives technology leaders a practical framework for understanding and addressing the issue.

It covers:

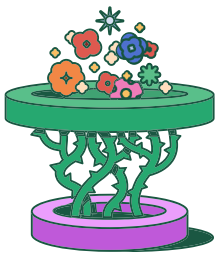
- The reason AI code sprawl feels overwhelming
- The real cost of wild code to your business
- How to assess your organization's exposure
- How to make IT an enabler, not a blocker

Ultimately, it's not about adding more restrictions or increasing workloads. It's about freeing every team – including IT – to work faster, more confidently, and more securely by default.

Naming the problem

AI code sprawl is already happening in most organizations

- `The recruiter` who builds an AI workflow to scrape data from candidate applications and assess them against job specs...without clarifying how that data is used or stored.
- `The RevOps analyst` who uses AI to pull data from your CRM and create weekly reports for stakeholders...leaving confidential financial information unsecured.
- `The HR coordinator` who creates an AI agent for employees to ask policy questions... and accidentally gives it (and other employees) access to sensitive compensation data.
- `The sales manager` who sets up a lead routing workflow with AI...then exits the company, leaving the workflow running with no owner or monitoring.



None of these employees opened a ticket, wrote anything to a repo, or informed IT – but all of them built something that now runs unattended and touches real data.

These aren't malicious actors. Employees are responding to top-down mandates to move faster with AI, and the barrier to entry has never been lower. Tools like Claude Code and Codex are making it easier for teams to build apps, agents, and automations without technical expertise or training.

Regardless of intention, the outcome is the same: organizations lack visibility into what software actually exists, who owns it, why it was created, and whether it can be safely modified or retired.

This phenomenon is sometimes known as **`wild code,`** because of how quickly it can spread and take root, faster than organizations can track, govern, secure, or maintain it.

The evolution of an existing problem

Technology leaders have encountered similar problems before:

- `SaaS sprawl:` the adoption of too many software-as-a-service products, which increases operational complexity for IT teams, introduces data silos that stall productivity, and wastes budget on barely-used tools.
- `Wild code changes the equation:` AI agents and tools can be built and deployed in minutes, outside established channels, by employees who don't even realize they've built something — let alone something that needs to be vetted, tested, and governed.
- `Shadow IT and AI:` the unapproved use of apps, devices, and AI for work, reducing IT's visibility and exposing organizations to data breaches, compliance and regulatory issues, and degraded end-user trust.

> “The population building software has changed. I’m seeing folks with little to no technical skills or software engineering backgrounds, build apps, agents, and agentic workflows. They aren’t ignoring the rules. The rules are written in a vocabulary nobody taught them.”
>
> BRAD RUMPH, FIELD CTO, TINES

The risks aren’t new. What’s changed is the rate of accumulation.

Traditional governance assumes that software development follows certain rules and processes. Wild code changes the equation: AI agents or tools can be built and deployed in a matter of minutes, building happens outside of established channels, and many employees don’t even realize they’ve “built” something – let alone something that needs to be vetted, tested, and governed.

The result? Every employee is a potential source of ungoverned code, making it exponentially harder for IT and security teams to keep up.

AI makes the building feel free, so the owning is invisible. Someone spins up a workflow to solve a problem, but they're not thinking about who patches it, what happens when an API changes, or where the data it touches ends up. This almost always lands on IT later if a workflow or connector breaks.

CHRISTINA TROY, SENIOR MANAGER, IT, TINES

The real cost of wild code

Wild code comes with significant operational, financial, and organizational costs.

Security risks

Wild code is expanding organizations' attack surface faster than security teams can address it.

IBM[1] found that security incidents involving shadow AI have more than doubled in the last year. Those incidents are also more expensive, with an average breach cost of $5.39M compared to the overall average of $4.99M. And one in five come with regulatory fines.

AI-generated code offers an entry point: **one study** by Veracode[2] found that almost half of all AI-generated code contains security flaws, while **another study by Escape**[3] revealed that more than 60% of "vibe-coded apps" contained vulnerabilities, with many containing high-impact vulnerabilities and secrets.

45% of AI-generated code contains security flaws.

VERACODE'S 2025 GENAI CODE SECURITY REPORT

These security costs aren't just financial. They also disrupt operations, damage end-user trust, and negatively affect business reputation, meaning organizations are often still paying for the incident long after it's resolved.

Security incidents involving shadow AI more than doubled to 43% in 2026, up from 20% in 2025.

In 21% of shadow AI security incidents, organizations reported paying a fine.

IBM'S 2026 COST OF A DATA BREACH REPORT

Financial waste

In addition to higher breach costs and compliance penalties, wild code also introduces more mundane financial waste. This includes:

- `Untracked AI spend` as employee-built apps and automations drive up ongoing AI/API costs with no clear owner to monitor usage
- `Unused seats or licenses` for existing software as employees turn to additional tools instead of approved solutions
- `Resource costs` as IT teams must track down and remediate wild code, reducing time available for more strategic, higher-value work

Over 40% of agentic AI projects will be canceled by the end of 2027, due to escalating costs, unclear business value, or inadequate risk controls.

GARTNER[4]

> "Where things get a little crazy is when individuals start building agents that duplicate the functionality of SaaS modules — simply because they don't like the way the workflow has been implemented in the SaaS tool they're already paying for!"
>
> MARK SETTLE, 7X CIO

Blocked innovation

Employees don't turn to shadow AI for no reason. In many cases, they have a real pain point – and feel they don't have the necessary tools or support to fix it.

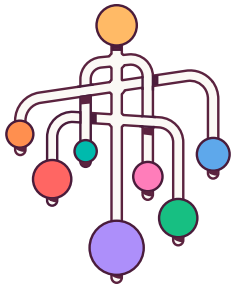
AI coding tools like Claude Code and Codex let subject matter experts build a solution based on their needs, without depending on additional technical resources. Instead of having to translate their problems to a developer or engineer and wait for availability, these tools let users address even their smallest problems and make meaningful improvements to their everyday work.

The innovation is real. The time savings, efficiency gains, and user experience improvements are valuable. It's the mechanism that's the issue.

Without an approved way for employees to make these optimizations, organizations must either block the innovation and stall progress, or accept the risks that come with wild code – both of which have real costs.

88% of executives believe employees have the tools they need, but only 21% of employees agree.

WALKME GLOBAL STUDY[5]



Increased burden on IT teams

A study into the effects of AI-assisted software development[6] found that while AI dramatically reduces the effort required to produce code, it shifts much of that effort downstream, to the people responsible for reviewing and understanding it.

This drains IT teams in multiple ways:

- `Misuses resources:` every ungoverned script someone has to track down, audit, or fix is time and expertise diverted from higher-value work
- `Increases technical debt:` scripts that aren't documented, tested, or reviewed add to a growing backlog of code that's fragile, hard to change, and riskier to keep running
- `Duplicates effort:` without visibility into what already exists, teams end up building the same tool in different silos

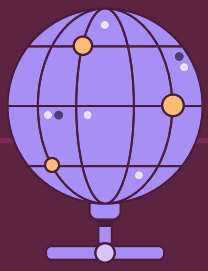
IT and security teams are already stretched thin. **Voice of Security 2026**[7] found that 76% of security professionals experienced burnout over the last 12 months, with heavy workloads and repetitive tasks surfacing as the key culprits. A **report by Auvik**[8] finds similar issues in IT - 60% of IT professionals reporting moderate or significant burnout.

IBM's research shows that AI and technology are being deployed faster than IT can track or govern: just 11% of tech leaders feel fully prepared for the scale of AI agent deployment expected over the next 12 months, and 77% of organizations report that AI adoption is already outpacing their governance capabilities[9]. If left unchecked, this growing pressure risks increasing burnout, which may lead to expensive errors or costly attrition.

> "The hidden costs of this phenomenon are the long-term support costs. If the agent drifts or delivers obviously erroneous results, who solves the problem, and how? If an agent is being used by multiple individuals, who approves changes?"
>
> MARK SETTLE, 7X CIO

“On a human level, the real weight isn’t any single incident – it’s the effort of staying on top of all of these pieces. It can be hard to keep a pulse when we need to know everything. Sometimes it means we don’t have time to make improvements to our own programs and processes. Burnout is real with this kind of fast-paced context switching.”

CHRISTINA TROY, SENIOR MANAGER, IT, TINES



Readiness checklist

Is wild code already inside your organization?

What's already been built

- [ ] We know exactly what non-technical teams have built with AI tools in the last 90 days
- [ ] If the employee who built a critical workflow left tomorrow, someone else in the organization would know it existed
- [ ] We have processes in place to translate existing AI-built apps into approved formats

Governance assumptions

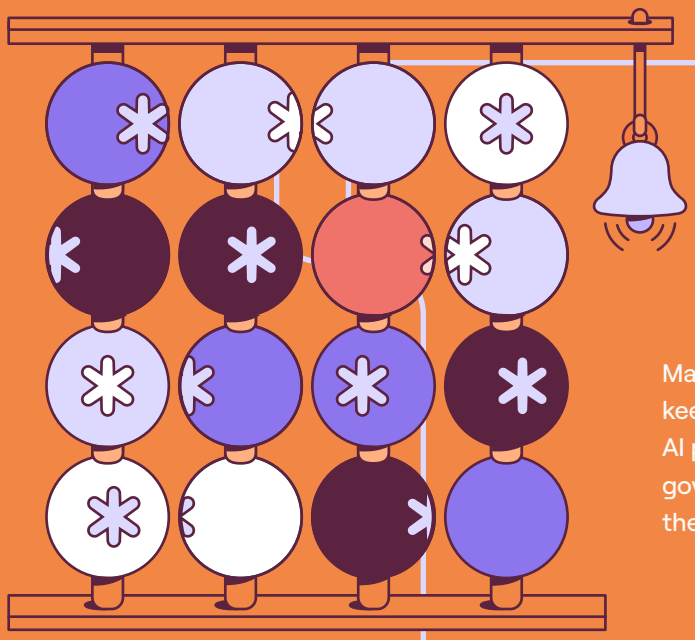
- [ ] Our AI policy is clear and accessible to all employees, not just those with pre-existing technical knowledge
- [ ] Employees outside of IT are aware they need to submit AI-assisted builds for review
- [ ] Our review process doesn't rely on employees knowing it exists; we have systems in place to ensure builds don't bypass it

Access and monitoring

- [ ] Only employees with relevant security training can connect sensitive data to an external tool right now
- [ ] Continuous monitoring allows us to discover issues within hours, not days or weeks – and not because something broke
- [ ] We know and control what data AI tools can connect to (including customer, business, and employee data)

What this means: Any unchecked box is an opportunity for wild code to take root and spread – even in organizations with strong governance in place.

Why the usual fixes fall short



Many organizations assume they can keep wild code under control through AI policies, mandatory training, or governance committees, but none of these address the root of the problem.

Employee education

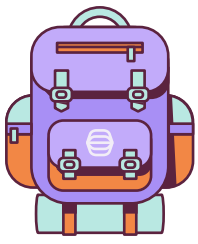
`What it is:` AI policies, awareness campaigns, and mandatory trainings

`Why it doesn't work:` The problem isn't simply a lack of knowledge.

The training gap is wider than most organizations think. For most non-technical employees, the rules are written in a vocabulary nobody taught them: concepts like production versus staging, secrets management, and least privilege.

Even when they do understand the risk, many employees are willing to take it. Deadlines, productivity expectations, and increased pressure to move faster provide ample motivation to build with AI, especially when poor visibility means mistakes can stay hidden.

Employee education assumes employees can recognize risk and make the right call, but rarely gives them the right incentive to do it.



"Most people don't yet have a mental model for secure, responsible AI use. Things like how to route a workflow webpage or connector to ensure the data remains within the secured workspace are not top of mind, especially when the AI will not generally guide you to secure it."

CHRISTINA TROY, SENIOR MANAGER, IT, TINES

Sandboxing

`What it is:` Isolated environments where employees can test models, prompts, and AI-generated code without exposing production systems or sensitive company data

`Why it doesn't work:` A sandbox creates a false sense of safety, because there's no governance once an experiment becomes operational.

Applications and automations only become truly useful when they access company data and operational systems. Sandboxes also can't account for actually running, monitoring, and maintaining code in production.

The sandbox label can create another problem, even for experienced users. When teams presume what they're building in a sandbox is safe, it can cause them to stop asking hard questions about it. By the time it leaves the protected environment, it hasn't been reviewed to the same standard as something built in the open.

> "Sandboxes solve the design problem and leave the runtime problem alone. They work for prototyping and evaluation, and they're weakest exactly where AI is most useful, which is against real schemas and real data."

BRAD RUMPH, FIELD CTO, TINES

AI steering committees

`What it is:` Dedicated governance committees that provide oversight for AI usage

`Why it doesn't work:` AI steering committees operate at a human pace, while wild code spreads at machine speed.

Steering committees worked well when code emerged through transparent processes and in established channels. But today, any employee can use AI to build applications and deploy them before a governance committee even knows they exist.

Governance built on predicting specific bad behaviors ages poorly, because those behaviors evolve faster than review cycles can handle. Committees may prohibit one risky practice only to find that employees have already adopted a new tool or discovered another route to the same outcome.

Many organizations also focus oversight on areas where strong controls already exist, like engineering, instead of applying those same governance practices to other functions.

From blocker to enabler

All of these approaches force IT teams into a gatekeeper role, seemingly 'shutting down' innovation and keeping IT locked in reactive mode. Think about it: how much of your team's time is spent being the 'no' in the room versus actually driving the business forward?

The fix isn't a better set of rules for people to follow or another layer of oversight. It's making the easiest path the governable one. When all employees can build safely and securely by default, everyone wins: teams are empowered to solve problems at scale, and IT becomes a true strategic enabler.

"The automation nobody can see is built by people in finance, operations, and HR, who don't think of themselves as developers. It never enters a repo, so code scanning will never find it."

BRAD RUMPH, FIELD CTO, TINES

The real solution to wild code

Governance by default

The answer isn't employee education, sandboxing, or AI steering committees. It's an environment where none of these are really needed.

To create this environment, CIOs can follow three core principals.

1. Govern proactively, not reactively:

   Instead of trying to catch risky builds after the fact, give employees an environment where governance is built in from the start and applied throughout the entire lifecycle – so both the environment and everything created within it are controlled by design, not by inspection.

   This way, all creation, execution, and monitoring takes place within a single platform, reducing maintenance overhead and giving IT complete oversight, while removing the need to switch between tools or search for hidden automations.

Think of it like this:

When weeds are threatening to take over your garden, you reach for a trowel or some weed killer. But what if you had a garden where weeds can't take root in the first place?

2. Make the safe path the only path:

Employees don't turn to wild code because they want to take risks or cause problems. They do it because they believe it's easier or faster than the "approved" option.

But there's no reason the approved option can't also be the easiest, fastest, and safest. Guide employees toward good sanctioned options, and use enterprise-tier admin controls to limit risky AI use within existing tools.

Take this a step further by ensuring the build experience in this environment is easy, quick to learn and delightful for the end user, so employees aren't tempted by other, unauthorized paths.

3. Trust AI for capability, own the controls around it:

AI is capable of great things, but only human judgment can decide what's safe for your organization.

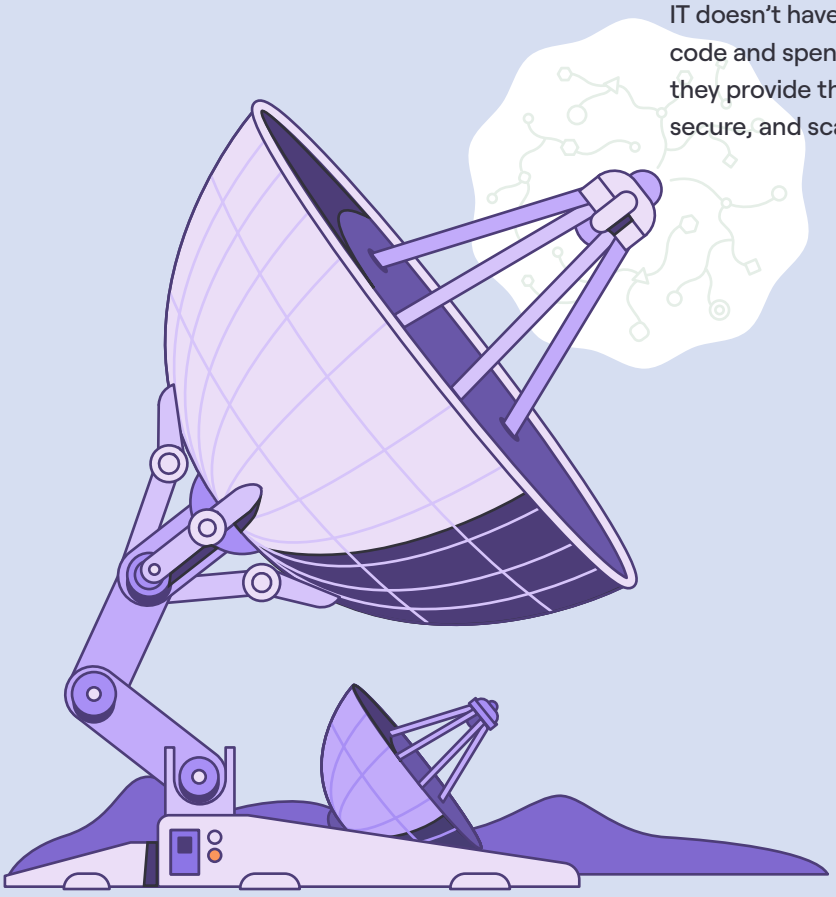
With one governed environment for all building, IT can implement guardrails by design. This means proactively setting the controls for that environment – like which data sources and business systems employees can connect to and what actions they're allowed to perform – to ensure that everything built within it automatically meets organization-wide requirements.

What this looks like in action

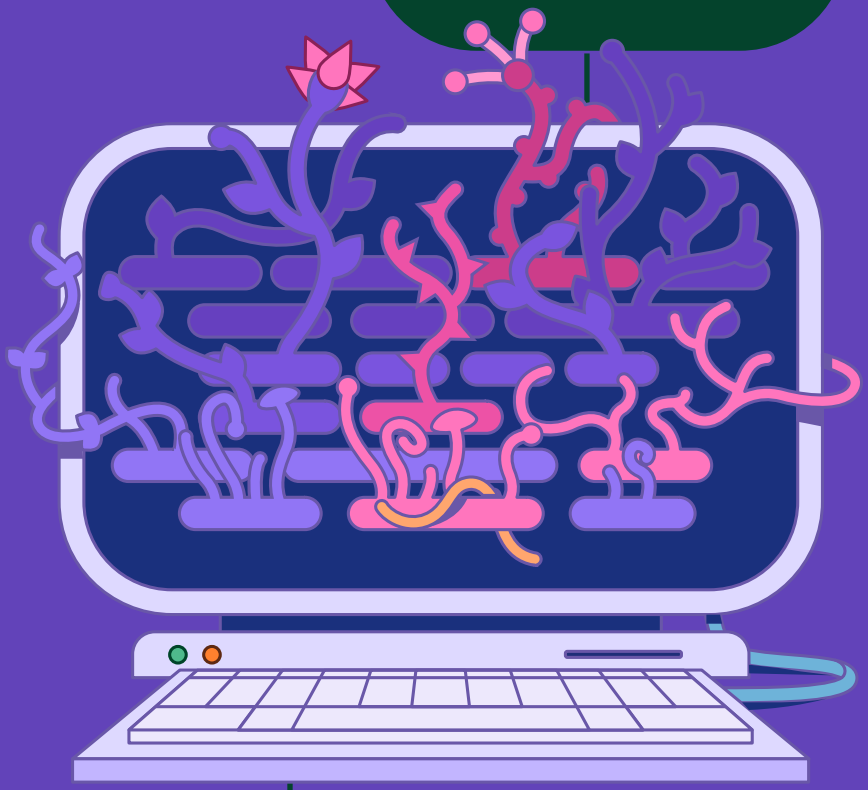
With the right environment, all building across the organization is:

- `Controlled:` only pre-approved data sources and actions are available, reducing risk and over-access
- `Unified:` IT has complete oversight and control of the entire building lifecycle, including what happens after something is built
- `Isolated:` each workflow step runs in isolation, and credentials are injected at runtime without being exposed
- `Monitored:` once a workflow is live, IT teams can monitor access, performance, dependencies, and AI spending
- `Transparent:` potential failures and proposed fixes remain visible for review

IT doesn't have to constantly scan for wild code and spend hours removing it. Instead, they provide the opportunity for real, secure, and scalable business growth.



Taming wild code for good



Empowering every employee to use AI to solve their problems is the differentiator between organizations that innovate quickly, securely, and at scale, and those still stuck in the weeds.

The answer isn't increasing the burden on IT teams, blocking innovation, or accepting risk. It's an environment where ungoverned building isn't possible in the first place.

That's why Tines built **Tines 3B.** Tines 3B is a single, secure environment for every team's important apps, agents, and automations. It gives people across the business the freedom to build with AI, while giving security and IT teams the control to run and govern that work safely – supporting organization-wide transformation and stopping wild code from taking root.

→ **Learn more and get started for free at** tines.com

RESOURCES CITED:

1. IBM's Cost of a Data Breach Report 2026
2. Veracode's GenAI Code Security Report
3. Escape's The State of Security of Vibe Coded Apps
4. Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027
5. WalkMe's Global Study
6. "An Endless Stream of AI Slop": How Developers Discuss the Burden of AI-Assisted Software Development
7. Voice of Security 2026
8. Auvik's IT trends 2025
9. IBM's 2026 tech leader study

tines