



How we made Trail of Bits AI-Native (so far)

Dan Guido, CEO | March 4, 2026 | [un]prompted

AI • PRODUCTIVITY

Thousands of CEOs just admitted AI had no impact on employment or productivity—and it has economists resurrecting a paradox from 40 years ago.

BY SASHA ROGELBERG | REPORTER

FEBRUARY 17, 2026, 1:32 PM ET

What does AI-native mean?

'We use AI' vs 'AI is on the team'

AI-assisted	People use AI tools to do their existing jobs faster (chatbots, copilots, summarizers)
AI-augmented	Workflows are redesigned around AI capabilities (agents in the loop, AI-first processes)
AI-native	The organization is architected from the ground up assuming AI is a core participant in delivery, operations, and how expertise compounds

At ToB: AI-native means our security expertise compounds.

Every project makes the next one faster, and every engineer operates with an arsenal of specialized agents built from 14 years of knowledge.

What people are actually resisting

01 Self-enhancing bias. A model of you outperforms you. The more senior and successful you are, the harder this hits. You believe in the intuition that got you here.

02 Identity threat. Security researchers identify deeply with their craft. AI that replaces skill feels like an attack on who you are.

03 Opacity. People think they understand human judgment and don't understand AI. In reality, they understand neither, but the feeling of not understanding kills the feeling of control.

04 Intolerance for imperfection. We forgive our own mistakes but not the machine's. One bad AI output confirms the bias. (Dietvorst et al.: people abandon algorithms after seeing them err once.)

The remedies that actually worked

Designing the system around the resistance, not against it

BARRIER	REMEDY	WHAT WE BUILT
Self-enhancing bias	Make it measurable: you can't argue you're already good enough	Maturity matrix with visible levels
Identity threat	Skill-allowing, not replacing: frame AI as making experts more dangerous	Skills repos, Maturity Matrix that rewards building (not just using), hackathons, not mandates
Intolerance for imperfection	Reduce the ways AI can fail embarrassingly, curate quality	Curated marketplace, sandboxing, guardrails, footgun reduction
Opacity	Write the rules so people know what's safe and approved	AI Handbook, usage policy, clear boundaries
(All of the above)	Make adoption visible and fast	Hackathons, copy-pasteable configs, CEO goes first

The “operating system” model

01 Pick a standard toolchain so you can support it.

02 Write the rules so risk conversations stop being ad hoc.

03 Create a capability ladder so improvement is expected, measurable, and rewarded.

04 Run tight adoption sprints so the org keeps pace with releases.

05 Package the learnings into reusable artifacts (repos, configs, sandboxes) so the system compounds.

Standardize on tools

☀ Coffee and Claude time?

How can I help you today?

+

Opus 4.6 Extended ▾ 🔊



Write



Learn



Code



Life stuff



Claude's choice

Write the rules

AI AT TRAIL OF BITS

This document provides comprehensive information about approved AI tools, services, and best practices for Trail of Bits team members. As AI capabilities continue to evolve rapidly, we've established guidelines to help you leverage these technologies effectively while maintaining our security and privacy standards.

To improve your chat prompts, take a short course in prompting from [LearnPrompting.org](#) or [Anthropic](#), review the Prompt Engineering Guide, and read [Anthropic's Claude Code Best Practices](#).

Join AI-Related Slack channels [#ml-random](#) [#ml-experiments](#) and [#wg-ml-assurance](#).

There is a twice-monthly AI/ML Reading Group on the company [Team Meetings calendar](#). The archive of papers, notes, and recordings is on Google Drive in the [AI/ML Reading Group folder](#).

USAGE POLICY

All services listed in this document are approved for use with sensitive data, with these important exceptions:

- **Cursor** cannot be used for client code, except for blockchain client code. Use Claude Code or Continue.dev instead.
- **Meeting recorders** cannot be used for client meetings when we're hired by a lawyer under privilege.

The table below is a convenient lookup for various AI services and their pre-approved use for different teams/work. If you have any questions about any usage of any specific tool or service, ask in Slack or ask your manager.

AI MATURITY MATRIX

Make it measurable

The specific tools and tasks below serve as examples and are by no means an exhaustive or requisite list for each role. Discuss how you use AI tools for tasks specific to your role with your manager to plot where you currently sit on the matrix.

ROLE	NOT ENGAGED (0) Not experimenting; no evidence of trying	CAPABLE (1) Experiments with tools personally	ADOPTIVE (2) Enthusiastic about new methods; actively shares the results with others	TRANSFORMATIVE (3) Invents new ways to use tools; maintains/builds tools
ASSURANCE	- Dismisses AI as hype or security risk - Refuses to use AI for any security work - Manually analyzes all vulnerabilities and writes reports from scratch - Believes AI can't understand security contexts	- Uses Claude for vulnerability explanations and report drafting - Uses Claude Code for code review, vulnerability analysis, and tool scripting - Uses Claude to summarize security papers and videos - Drafts blog posts using our Claude Project with Tech Clarity Expert style	- Maintains custom agent instructions (CLAUDE.md) - Contributes to Wilson, our internal AI auditing tool - Active in #ml-experiments, testing and verifying AI strategies - Uses NotebookLM to synthesize multiple audit reports	- Builds agent systems with Claude Agent SDK; runs agents that autonomously produce audit artifacts — findings, triage, report drafts - Develops systematic approaches to prompt optimization (e.g., DSPy, custom frameworks) - Designs workflows where agents execute full analysis passes, not just answer questions - Leads AI knowledge-sharing initiatives (Reading Group, workshops, team training) - Publishes blog posts on AI for security research
ENGINEERING	- Calls AI coding assistants "too risky" - Has never tested AI-generated code - Relies only on Stack Overflow snippets - Dismisses colleagues who use AI tools	- Uses Claude Code for code generation and explanation - Generates boilerplate with Claude Code - Can explain and debug code with AI assistance - Follows AI best practices from handbook	- Uses Claude Code subagents effectively - Maintains custom CLAUDE.md across projects - Skilled at writing project requirements docs and using planning mode - Contributes to building MCP servers	- Builds agent systems with Claude Agent SDK; runs autonomous agents that ship PRs, close issues, and benchmark results without hand-holding - Develops systematic approaches to prompt engineering (e.g., DSPy, evals) - Fine-tunes models or builds evals for security use cases - Develops AI-enhanced security tools (fuzzers, analyzers) - Designs agent workflows that produce multiple outputs from the same engineer
SALES	- Writes proposals and SOWs from old templates without AI input - Manually compiles weekly pipeline reports from HubSpot - Prepares for client calls by reading old emails - Tracks account activity and renewal timelines in spreadsheets	- Drafts proposals, SOWs, and client emails with Claude - Uses Claude to research prospects and prep for discovery calls - Summarizes meeting notes with Granola for call follow-ups and action items - Uses Claude to draft QBR decks and account review materials	- Creates Claude Projects for active deals with stakeholder maps and account context - Uses Claude Cowork with Granola, Slack, Gmail, and HubSpot MCP servers connected - Uses job-brand skills for competitive positioning and proposal differentiation - Builds reusable SOW and proposal templates that cut drafting time significantly	- Has agents producing recurring deliverables — pipeline reports, QBR prep, account reviews — autonomously - Creates AI-driven systems for conference prep: pre-booking meetings, enriching attendee lists, scoring leads - Builds tools the GTM team uses daily — SOW generators, competitive intel dashboards, capacity analyzers - Designs agent workflows that turn portfolio management from reactive deal-closing into systematic account growth
PROJECT MANAGEMENT & OPERATIONS	- Refuses to use AI for "sensitive" client work - Takes manual meeting notes only - Dismisses AI tools as making PMs lazy - Manually handles every ticket and task	- Drafts project status updates and client emails with Claude - Uses Granola for meeting notes and action item tracking - Summarizes client requirements and SOWs with Claude - Uses Claude to prep for kickoff calls and client check-ins	- Creates Claude Projects for active client engagements with project context - Uses Claude Cowork with Granola, Slack, and Gmail MCP servers connected - Automates QBR prep and status reporting with reusable prompts - Shares effective workflows and templates with the PM team	- Designs end-to-end AI-powered client lifecycle - Creates automated client reporting systems - Designs workflows where agents handle the information processing so PMs focus on client relationships - Has agents executing recurring operational work — status reports, QBR prep, client updates — autonomously



Create an adoption engine (Hackathons)

Done (Q4 2025) ▾

○ Claude Code Hackathon

+ Add

We're looking to resolve as many issues as possible in our most recognized / most critical PUBLIC repositories. Go to our Github org and list all public repositories sorted by number of stars, pick one, and dive in.

Before starting, make sure you fully set up your Claude Code. Review the AI Handbook, particularly for recommended user-level settings in `~/ .claude` :

 [AI Handbook](#)

We'll measure your success in this hackathon via *activity*. We want to see everyone make the largest number of changes in issues (filed or fixed), PRs (reviewed, deconflicted, merged), branches (created or deleted), etc. in each codebase. We're not looking at lines of code.

Set up to work in pairs. Every change should get reviewed by your buddy.

Dan Guido Feb 10th at 02:47

Claude Code Hackathon v2: Autonomous Agents

Last November we got everyone comfortable with Claude Code by cleaning up our public repos. Since then, Claude Code has shipped [skills](#), [plugins](#), [hooks](#), [MCP servers](#), [native sandboxing](#), [agent teams](#), and a full [Agent SDK](#). Most of you haven't used any of it. Time to fix that.

The objective: Ship the most impactful changes across our AI toolchain and public repositories.

The twist: Engineers **must** work in **bypass permissions mode** (fully autonomous agent, no approve-every-action). You'll use our [devcontainer](#), [dropkit](#), or the [native macOS sandbox](#) to do this safely.

Three tracks:

1. *Engineers (Adoptive or Capable on AI Maturity):* Pick a public repo, fix real issues, file real PRs. Same as last time, but you're running Claude autonomously. At our demo session, share your setup and one technique that worked for achieving autonomous coding.
2. *Non-engineers:* Build a reusable Claude Code skill or set up MCP integrations for your daily tools (Slack, Gmail, Asana, Google Workspace). No more one-off Slack bots. Everything you build goes into our shared skills repo so the whole team can use it.

Before the hackathon, everyone must:

1. Review and [help update](#) our [AI Handbook](#)
2. Set up your sandbox (engineers) or MCP servers (non-engineers)
3. Configure your personal `~/ .claude/CLAUDE.md`
4. Install the ToB plugin marketplaces ([internal](#), [external](#), and [curated](#))
5. Pick a partner and a target, then enter your info in the [tracking spreadsheet](#)

Dates: Thursday, February 19 and Friday, February 20

Channel: [#company-dev](#) (all hackathon activity here)

Demos: At the next town hall and lunch & learn, 5 minutes each, everyone presents

Questions, ideas, or feedback: drop them in the thread. I'm especially interested in:

- Which repos need the most love?
- What skills would be most useful to build for non-engineers?
- Who wants to lead the Track 3 projects?

Capture the work as reusable artifacts (Skills repos)

Available Plugins

Smart Contract Security

Plugin	Description
building-secure-contracts	Smart contract security toolkit with vulnerability scanners for 6 blockchains
entry-point-analyzer	Identify state-changing entry points in smart contracts for security auditing

Code Auditing

Plugin	Description
audit-context-building	Build deep architectural context through ultra-granular code analysis
burpsuite-project-parser	Search and extract data from Burp Suite project files
differential-review	Security-focused differential review of code changes with git history analysis
insecure-defaults	Detect insecure default configurations, hardcoded credentials, and fail-open security patterns
semgrep-rule-creator	Create and refine Semgrep rules for custom vulnerability detection
semgrep-rule-variant-creator	Port existing Semgrep rules to new target languages with test-driven validation
sharp-edges	Identify error-prone APIs, dangerous configurations, and footgun designs
static-analysis	Static analysis toolkit with CodeQL, Semgrep, and SARIF parsing
testing-handbook-skills	Skills from the Testing Handbook : fuzzers, static analysis, sanitizers, coverage
variant-analysis	Find similar vulnerabilities across codebases using pattern-based analysis

Safe third-party adoption (Curated marketplace)

[README](#)[CC-BY-SA-4.0 license](#)[Security](#)

Curated Skills Marketplace

Trail of Bits' reviewed and approved Claude Code plugins. Every skill and marketplace here has been vetted for quality and safety.

Why This Exists

We don't want people at Trail of Bits installing random plugins from GitHub repos we haven't reviewed. Published skills have been found with backdoors and malicious hooks, and the ecosystem has no built-in quality gate. This repo is how we solve that problem internally.

Everything here has been code-reviewed by Trail of Bits staff. We're sharing it publicly so the broader community benefits from the same vetting.

Make defaults copy-pasteable (Claude Code configs)

Trail of Bits Claude Code Config

Opinionated defaults, documentation, and workflows for Claude Code at Trail of Bits. Covers sandboxing, permissions, hooks, skills, MCP servers, and usage patterns we've found effective across security audits, development, and research.

Also see: [skills](#) · [skills-curated](#) · [claude-code-devcontainer](#) · [dropkit](#)

First-time setup:

```
git clone https://github.com/trailofbits/claude-code-config.git
cd claude-code-config
claude
```

Then inside the session, run `/trailofbits:config`. It walks you through installing each component, detects what you already have, and self-installs the command so future runs work from any directory. Run `/trailofbits:config` again after updates.

Contents

[Getting Started](#)

- [Read These First](#)
- [Prerequisites](#)
- [Shell Setup](#)

Sandbox by default (Devcontainer + Dropkit)

dropkit

A command-line tool for managing DigitalOcean droplets with automated setup, SSH configuration, and lifecycle management.

Features

- 🚀 **Quick droplet creation** with cloud-init automation
- 🔑 **Automatic SSH configuration** - just run `ssh dropkit.<droplet-name>`
- 🏠 **Tailscale VPN** - secure access via Tailscale (enabled by default)
- 👤 **User management** - automatically creates your user account on droplets
- 🏷️ **Smart tagging** - organizes droplets by owner for easy filtering
- 🛡️ **Security-first** - SSH key validation, confirmation prompts for destructive operations
- 📊 **Rich CLI** - beautiful tables, progress indicators, and helpful error messages
- 🔄 **Complete lifecycle** - create, list, resize, destroy droplets with ease
- 🛌 **Hibernate/Wake** - snapshot and destroy to save costs, restore later with one command

Claude Code in a devcontainer

A sandboxed development environment for running Claude Code with `bypassPermissions` safely enabled. Built at [Trail of Bits](#) for security audit workflows.

Why Use This?

Running Claude with `bypassPermissions` on your host machine is risky—it can execute any command without confirmation. This devcontainer provides **filesystem isolation** so you get the productivity benefits of unrestricted Claude without risking your host system.

Designed for:

- **Security audits:** Review client code without risking your host
- **Untrusted repositories:** Explore unknown codebases safely
- **Experimental work:** Let Claude modify code freely in isolation
- **Multi-repo engagements:** Work on multiple related repositories

Reduce footguns (package managers)

developer-tools

Security hardening for developer machines.

configure-package-manager-defaults.sh

Deploys secure defaults for npm, pnpm, pip, Go, and Ruby to protect against supply chain attacks.

What it configures:

Setting	Location	Effect
<code>ignore-scripts=true</code>	Global npmrc (detected dynamically)	Blocks postinstall scripts (main malware vector)
<code>save-exact=true</code>	Global npmrc	Pins exact versions, prevents surprise upgrades
<code>prefer-binary=true</code>	<code>/Library/Application Support/pip/pip.conf</code>	Prefers wheels over source distributions
<code>GOPROXY=proxy.golang.org,off</code>	Per-user via <code>go env -w</code>	Forces proxy, blocks direct VCS fetches
<code>GOSUMDB=sum.golang.org</code>	Per-user via <code>go env -w</code>	Verifies checksums against transparency log
<code>gem: --no-document</code>	System gemrc (detected dynamically)	Skips doc generation, enforces HTTPS

Note: Config paths are detected dynamically. npm uses `npm config get globalconfig` (Homebrew: `/opt/homebrew/etc/npmrc`). Ruby uses `gem environment` (macOS: `/Library/Ruby/Site/gemrc`). pnpm reads the same npmrc file.

Connect with real tools (slither-mcp)

Slither MCP Server

 python 3.11+ License AGPL v3

A Model Context Protocol (MCP) server that provides static analysis capabilities for Solidity smart contracts using [Slither](#).

Overview

This MCP server wraps Slither static analysis functionality, making it accessible through the Model Context Protocol. It can analyze Solidity projects (Foundry, Hardhat, etc.) and generate comprehensive metadata about contracts, functions, inheritance hierarchies, and more.

You can also use Slither MCP as an easy-to-use Slither API for other use cases.

Features

- **Caching:** Slither runs are cached to `{PROJECT_PATH}/artifacts/project_facts.json` for faster subsequent loads
- **MCP Tools:** Query contract and function information through MCP tools
- **Security Analysis:** Run Slither detectors and access results with filtering
- **Comprehensive Analysis:** Extracts detailed information about:
 - Contract metadata (abstract, interface, library flags)
 - Function signatures and modifiers
 - Inheritance hierarchies
 - Function call relationships (internal, external, library calls)
 - Security vulnerabilities and code quality issues
 - Source code locations

Results (so far)

What changed after building the system

Tooling scale (across internal + public repos):

- 94 plugins, 201 skills, 84 agents, 29 commands, 125 scripts
- 414 reference files encoding domain expertise
- Covers the full consulting lifecycle: sales, proposals, audit setup, 20+ vulnerability-class-specific analyzers, fuzzing, report writing, delivery, project management, recruiting, onboarding

Delivery impact:

- For certain engagements: from ~15 bugs/week to ~200 bugs/week
- ~20% of all bugs reported to clients are now initially discovered by AI
- Sales team averaging \$8M rev per rep (industry benchmark: \$2-4M)

The replicable recipe

01 Standardize on one agent workflow you can support.

02 Write an AI Handbook so risk decisions aren't ad hoc.

03 Create a capability ladder so improvement is expected.

04 Run short adoption sprints that force hands-on usage.

05 Capture everything as reusable artifacts: skills + configs + curated supply chain

06 Make autonomy safe with sandboxing + guardrails + hardened defaults

Open questions

What we haven't solved yet

1. Private inference: Can open models replace closed ones?
 - a. Local models (Qwen3-Coder-Next) are interesting but not good enough
 - b. We're evaluating on-prem inference servers to run 230B+ models at full precision
 - c. Speed matters more than size. No one uses a slow model, even if it's smart.
 - d. In the meantime: private inference providers like Tinfoil.sh are getting compelling.
2. Prompt injection and client code protection
 - a. How do we prevent agents from exfiltrating client code? We're lucky so many ToB clients are OSS
 - b. Today: Our primary internal audit tools can't touch the web
 - c. Next (maybe): agent-native shells like nono and agentsh that enforce policy at the syscall level
3. Policy enforcement and continuous learning at scale
 - a. How to turn the company's usage into a feedback loop that continuously improves the OS
 - b. Next (maybe): a master MCP server with every internal resource that enforces policy

References

USE OUR GITHUB REPOS!

- [trailofbits/skills](#)
- [trailofbits/skills-curated](#)
- [trailofbits/claude-code-config](#)
- [trailofbits/claude-code-devcontainer](#)
- [trailofbits/dropkit](#)
- [trailofbits/slither-mcp](#)

TWEET AT US!

- [@dguido](#) + [@trailofbits](#)
- Email: dan@trailofbits.com

[www.trailofbits/careers](https://www.trailofbits.com/careers) – come work with us