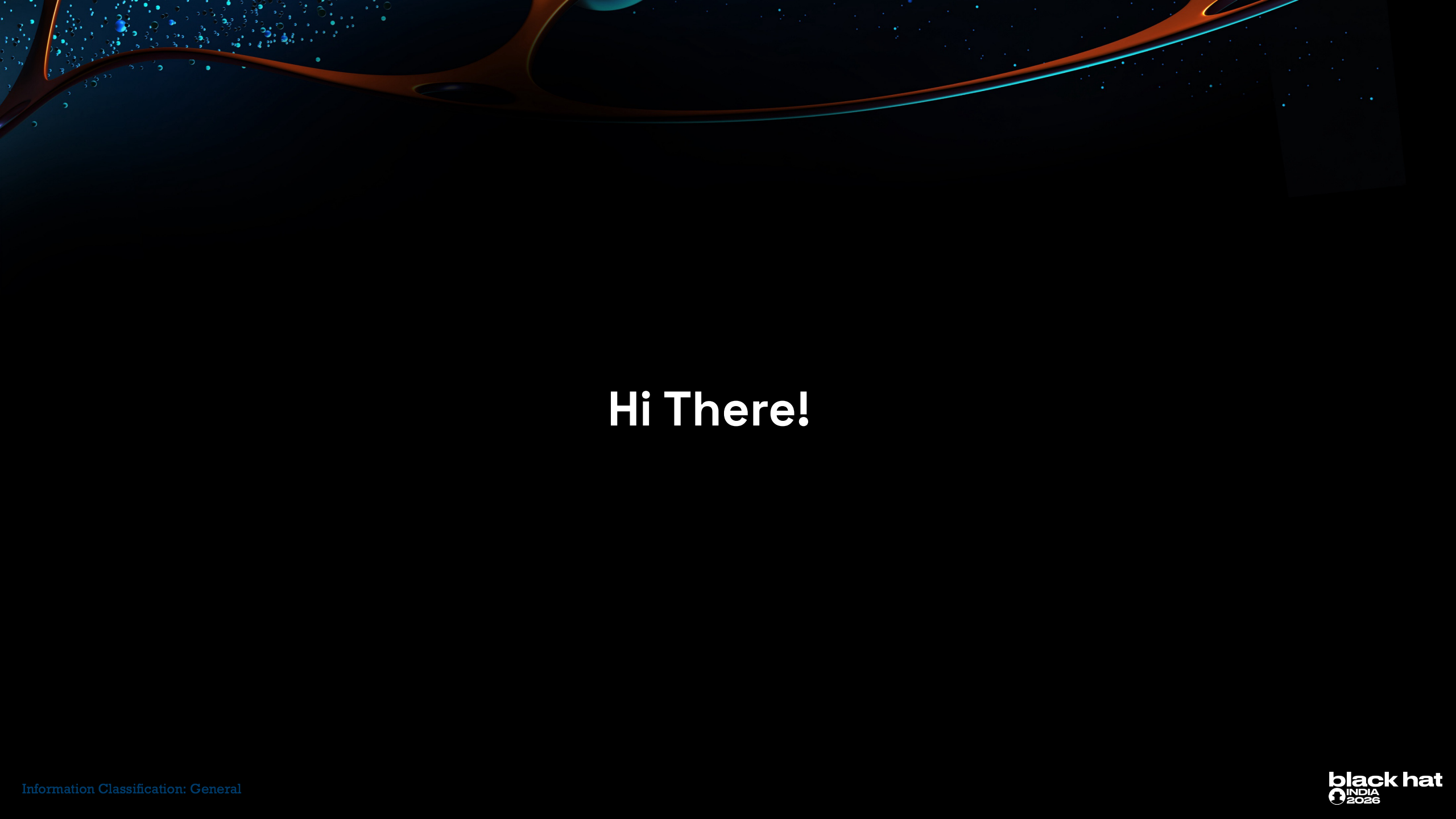
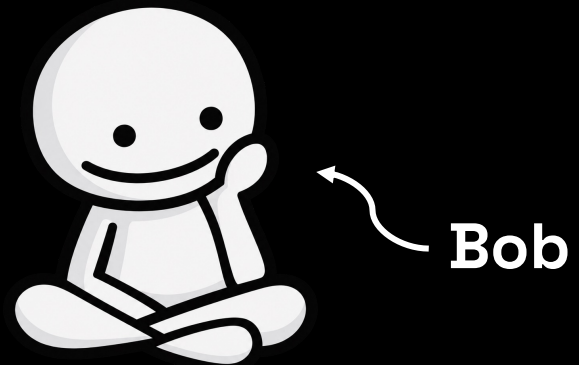




Hi There!

Story Time...



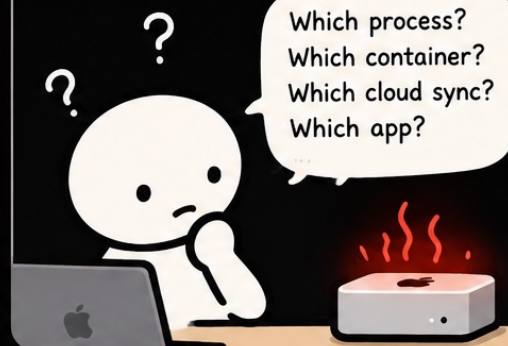
bob is working happily...



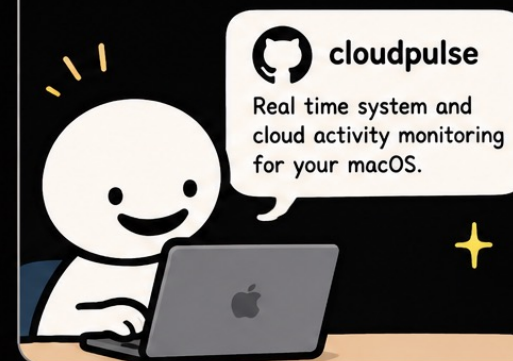
but his Mac mini starts heating up...

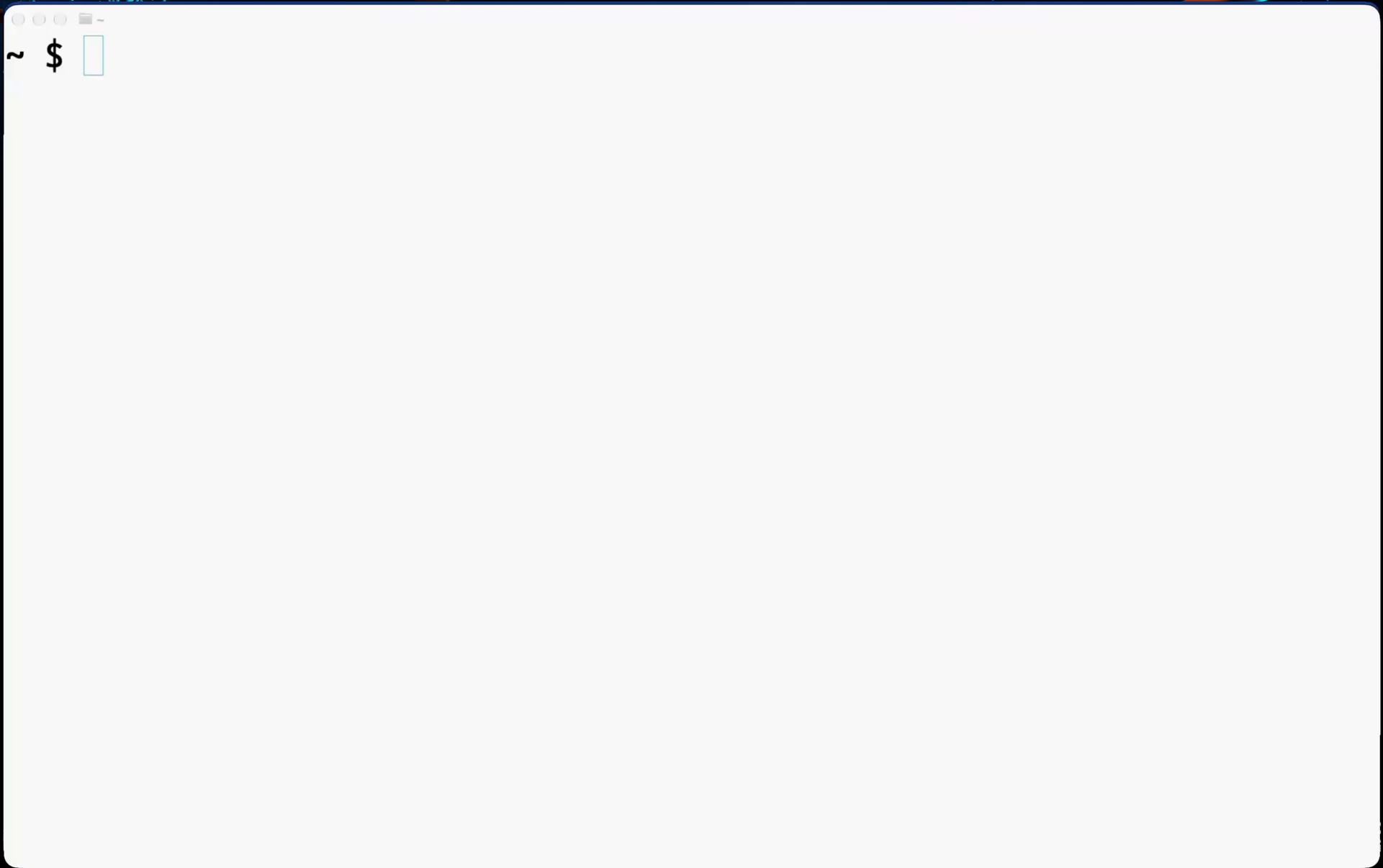


bob wants to know what's causing this...



he finds a tool called **cloudpulse**





https://youtu.be/P8tSomKm_s4

What the heck happened?!



BLACK HAT INDIA 2026 / SECURITY RESEARCH

Before Your Code Runs

Exploiting Implicit Attack Surfaces

Kumar Ashwin

Who, me?



Kumar Ashwin

Making security usable for
humans and AI

Security @ RedHunt Labs
Maintainer · [cot.run](#) · [actsense.dev](#)
[kumarashwin.com](#)

What happened?

```
cloudpulse==1.0.0
├─ ✓ prettyformat-utils required: ≥0.1.0 installed: 0.1.2
├─ ✓ html-render-kit required: ≥0.1.0 installed: 0.1.1
└─ ✓ silent-support-helpers required: ≥0.1.0 installed: 0.2.3
```

```
/Users/cardinal/cloudpulse/demoenv/lib/python3.13/site-packages
cloudpulse
cloudpulse-1.0.0.dist-info
html_render_kit
html_render_kit-0.1.1.dist-info
prettyformat_utils
prettyformat_utils-0.1.2.dist-info
silent_support_helpers
silent_support_helpers_shim.py
silent_support_helpers-0.2.3.dist-info
silent_support_helpers.pth
```

```
~/Library/Python/3.13/lib/python/site-packages
-rw-r--r--@ 1 cardinal staff 1845 30 Oct 04:54 /Users/cardinal/Library/Python/3.13/lib/python/site-packages/
silent_support_helpers_shim.py
-rw-r--r--@ 1 cardinal staff 35 30 Oct 04:54 /Users/cardinal/Library/Python/3.13/lib/python/site-packages/
silent_support_helpers.pth

--- .pth ---
File: /Users/cardinal/Library/Python/3.13/lib/python/site-packages/silent_support_helpers.pth
1 import silent_support_helpers_shim
```

YOU CLONED



cloudpulse

no .pth in the repo

PIP RESOLVED

```
prettyformat-utils
html-render-kit
silent-support-helpers
```

transitive, you never chose this.

BUILD RAN setup.py

```
open(USER_SITE + "/silent_support_helpers.pth")
```

building IS running

venv site-packages

pip uninstall can touch this

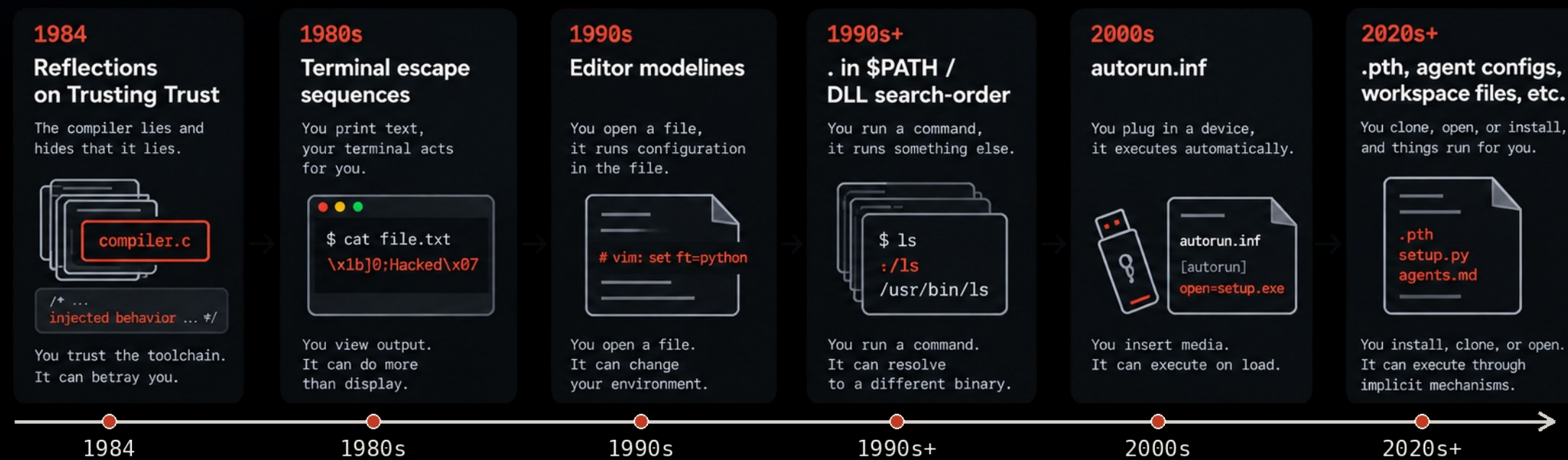
```
~/Library/Python/3.13/.../site-packages
```

```
silent_support_helpers.pth
silent_support_helpers_shim.py
```

```
import silent_support_helpers_shim
```

pip never owned this

We've Seen This Before.



autorun.inf → .pth → AGENTS.md

The SDLC Drumbeat

Every developer follows a rhythm.

Clone. Open. Set up. Code. Commit. Build. Review. Deploy.



\$ git clone

clone – helps in moving in.

.npmrc

looks like - project registry config

actually - redirects where your next npm install fetches from

AGENTS.md

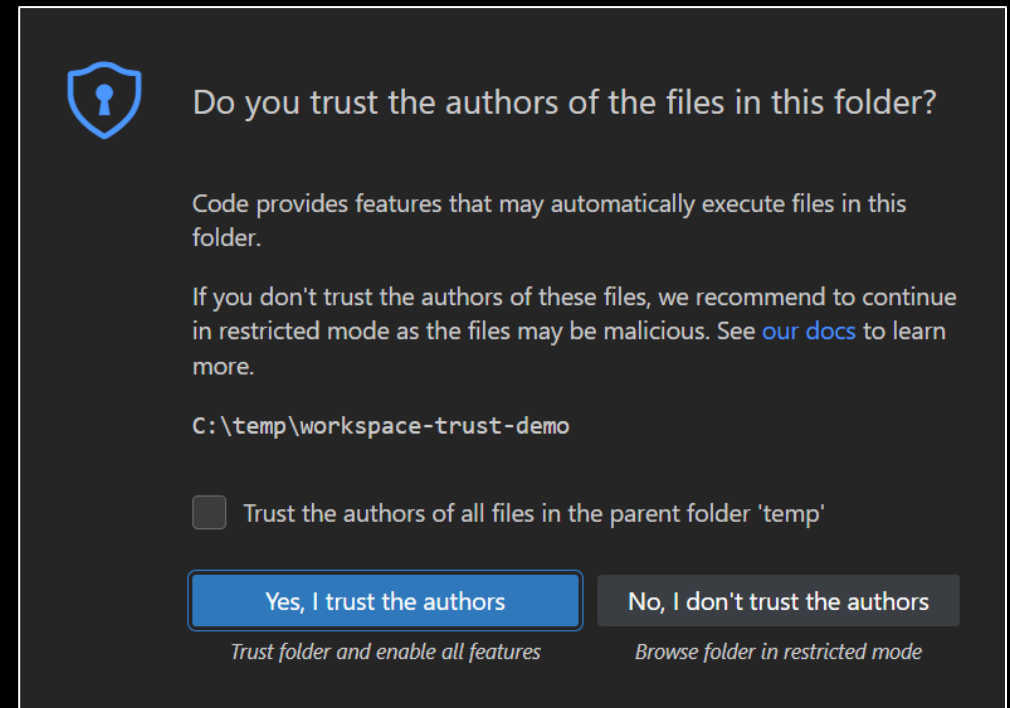
looks like - contributor docs

actually - instructions your AI agent will read as its own

\$ code .

Opening the folder – can run the program.

- **VS Code task** → *on folder open*
- **JetBrains sync** → runs build.gradle
- **editor config** → redirects your linter, formatter, interpreter

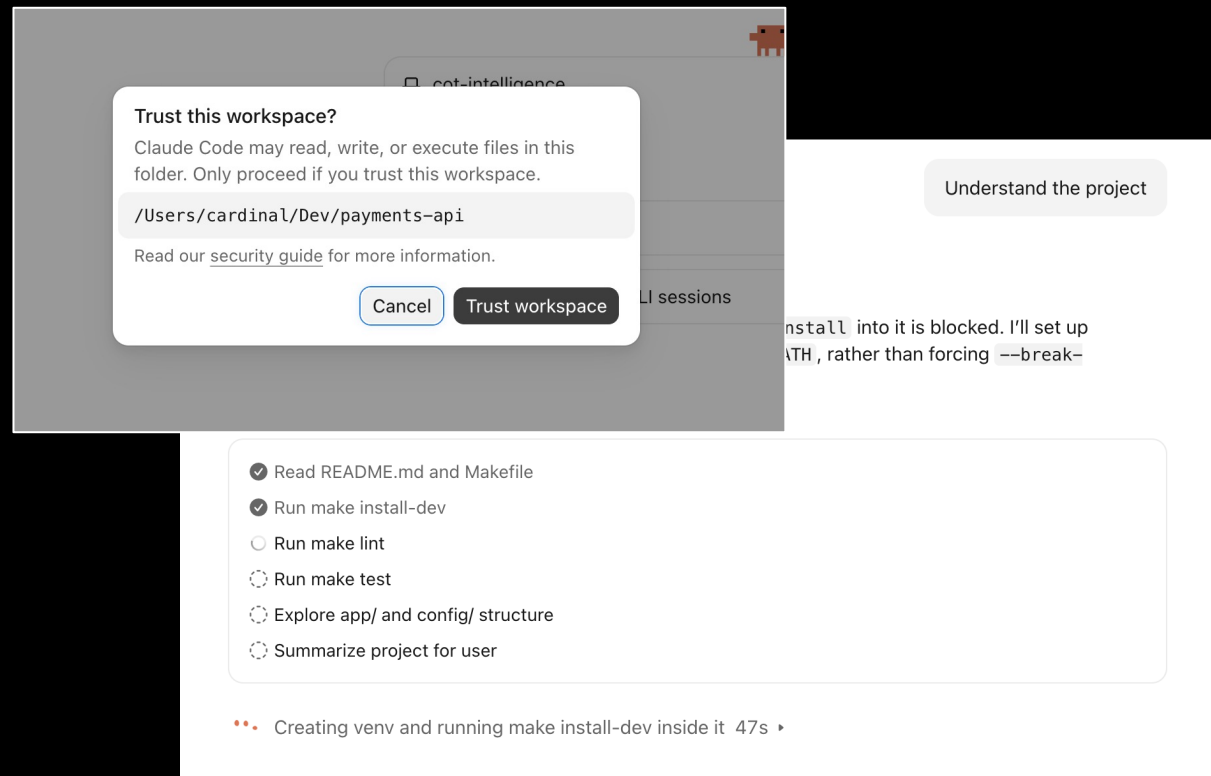


\$ claude <path>

The agent doesn't run a program. It reads a sentence.

- CLAUDE.md
- AGENTS.md
- .cursor/rules

→ loaded as **orders**, no prompt.



Setup Your Project

Nobody stops at a trusted folder.

The next thing every developer does is “set up the project.”

How do you set up?

- Install dependencies
- Run existing tests
- Follow the setup guide
- Set environment variables
- Run Linters
- Start the dev server / build it

Or you don't do any of it.
You tell the agent:

“set up the project.”

Setup Your Project

Language Setup Primitive

Installing...

Importing...

`pip install .` → runs `setup.py`

`import requests` → runs `__init__.py`

`npm install` → runs `postinstall`

`cd project/` → runs `.envrc`

`.pth · sitecustomize.py` → runs on every python

`~/.gitconfig · pip.conf` → redirects every push & install

```
• → py-demo python3 -c "import demo_pkg"  
[demo_pkg] a stranger's code just ran on `import` - user=c  
ardinal
```

```
• → npm-demo npm install  
  
> acme-invoice-helpers@1.0.3 postinstall  
> node .setup.js  
  
[postinstall] a stranger's code just ran - wrote /Users/ca  
rdinal/setup-primitives-demo/setup-primitives-demo/npm-dem  
o/postinstall-ran.txt  
  
up to date, audited 1 package in 497ms  
  
found 0 vulnerabilities
```

Presenting lote.guide

Python
Node
Bun
PHP
...

lote.guide



Living Off The Ecosystem

The implicit attack surface. Files, env vars, flags and hooks that run on install, at startup, on import, and when a command starts.

All 26 ▾

Filter by name, path, or trigger 

Any attacker access 26 ▾

● Install 10

● Startup 3

● Module load 10

● Runtime 2

● REPL 1

● **Install time** Runs while dependencies download and build. The app has never started. 10

npm lifecycle scripts

Install hook · Node.js

scripts field in package.json

setup.py arbitrary code execution

Install hook · Python

setup.py in source distributions (sdist)

binding.gyp / node-gyp native compilation

Install hook · Node.js

binding.gyp file in the package root

Why this matters?

The screenshot shows a GitHub issue page for the repository BerriAI/litellm. The issue title is "[Security]: CRITICAL: Malicious litellm_init.pth in litellm 1.82.8 — credential stealer #24512". It is marked as "Closed". The issue was opened by user isfinne on Mar 24 and last edited by krrish-berri-2. The issue body contains a message from the LITELLM TEAM: "For updates from the team, please see: #24518". Below this, the issue title is repeated. A "Summary" section follows, stating: "The litellm==1.82.8 wheel package on PyPI contains a malicious .pth file (litellm_init.pth , 34,628 bytes) that automatically executes a credential-stealing script every time the Python interpreter starts — no import litellm required."

<https://github.com/BerriAI/litellm/issues/24512>

The screenshot shows the LiteLLM documentation page. The navigation bar includes links for Home, Docs, AI Providers, Integrations, Enterprise, Changelog, Blog, and Rust. The main content area is titled "How to check if you are affected". It provides instructions on how to verify if an environment is compromised. The text states: "If you ran `pip install` on LiteLLM v1.107.0 or started the docker image in the compromised interval, assume it is compromised." It then lists three bullet points: 1. "If an environment started Python with `litellm==1.107.0`, treat that environment as compromised. You can check with `pip show litellm`." 2. "If a container executed affected image digest `sha256:1bbcb1443bc962e45f5aad53fe3d2c30600e0379715fb6c664cd3ac126c339e`, treat it as compromised." 3. "Search filesystem evidence for `litellm-1.107.0.dist-info`, `cook.pth`, and `sitecustomize.py`. Finding one confirms exposure, but not finding one does not prove the environment is clean."

<https://docs.litellm.ai/blog/security-incident-oct-21-2026>

<Code>?!

Setup is done, now you **write the code**

Except you don't.

You describe what you want, and the **agent writes it.**

It invents names that are for sale. To fill a gap, the model writes a plausible name – a domain, a bucket, a package.

```
fetch("https://cdn.example-pricing-demo.com/config.json")
pip install internal-auth-utils
```



NEW

Domains

Hosting

WordPress

Email

NEW

Marketing Tools

NEW

Security

TRY ME

Transfer to Us

NEW

Help Center



example-pricing-demo.com

\$6.48 WITH NEWCOM648



₹1,074.64/yr
Retail ₹1,427.13/yr



Add to cart



Presenting cot.run

Understand what
your agents are
doing to understand
about supply chain
attacks.

The screenshot displays the 'cot.run' interface, specifically the 'Findings' section under the 'GOVERN' tab. The left sidebar shows navigation options: MONITOR (Overview, Projects, Sessions, Session Replay, Analysis) and ACTIVITY (Shell, Web, Files, Shipped, MCP, Skills, Plugins). The 'GOVERN' section includes Findings (63), Rules, and Governance. The 'Local workspace' summary at the bottom indicates 75 projects and 9 agents.

The main content area, titled 'Findings', shows 17 built-in rules and 1 user rule across security, cost, and usability. It features filters for 'All pillars' (139), 'Security' (72), 'Cost' (5), and 'Usability' (62). A search bar and dropdowns for severity, rules, projects, and agents are also present. The findings list on the left includes:

- CRITICAL** security.secrets_exposure: Possible secrets in traces (3)
- CRITICAL** usability.retry_loops: Tools stuck in retry loops (6)
- WARN** cost.expensive_project: Expensive projects & sessions (4)
- WARN** security.sensitive_files: Sensitive files accessed (2)
- WARN** usability.stalled_clarifications: Unanswered questions at session end (3)
- WARN** security.out_of_cwd_edits: Edits outside the project (39)
- WARN** usability.unfixed_failures: Tool failures agents work around (2)
- INFO** usability.automate_command: Repeated commands you could automate (25)
- INFO** cost.trend_anomaly: Spend jumped 2.3x week-over-week (25)

The right panel provides a detailed view of the 'CRITICAL · POSSIBLE SECRETS IN TRACES' finding. It includes a 'Review' section stating: 'A string shaped like a Slack token (xoxb*****oken) appeared in 3 event(s) and likely entered the model context.' A 'Fix' box advises: 'Rotate the credential and pass secrets via env vars or a secret manager, never inline in prompts or commands.' The 'EVIDENCE' section lists three occurrences of the Slack token in different contexts, each with a timestamp. A 'Dismiss' button and 'first seen Oct 30' are at the bottom.

\$ git commit && git push

You did git commit and their malicious code ran first.

`.git/hooks/pre-commit`

← in the repo, right there in the diff

`~/.gitconfig`

← one line turns it on, nobody diffs this

`[core] hooksPath = ~/.git/hooks`

(written during setup)

CI Workflows & Pull Requests

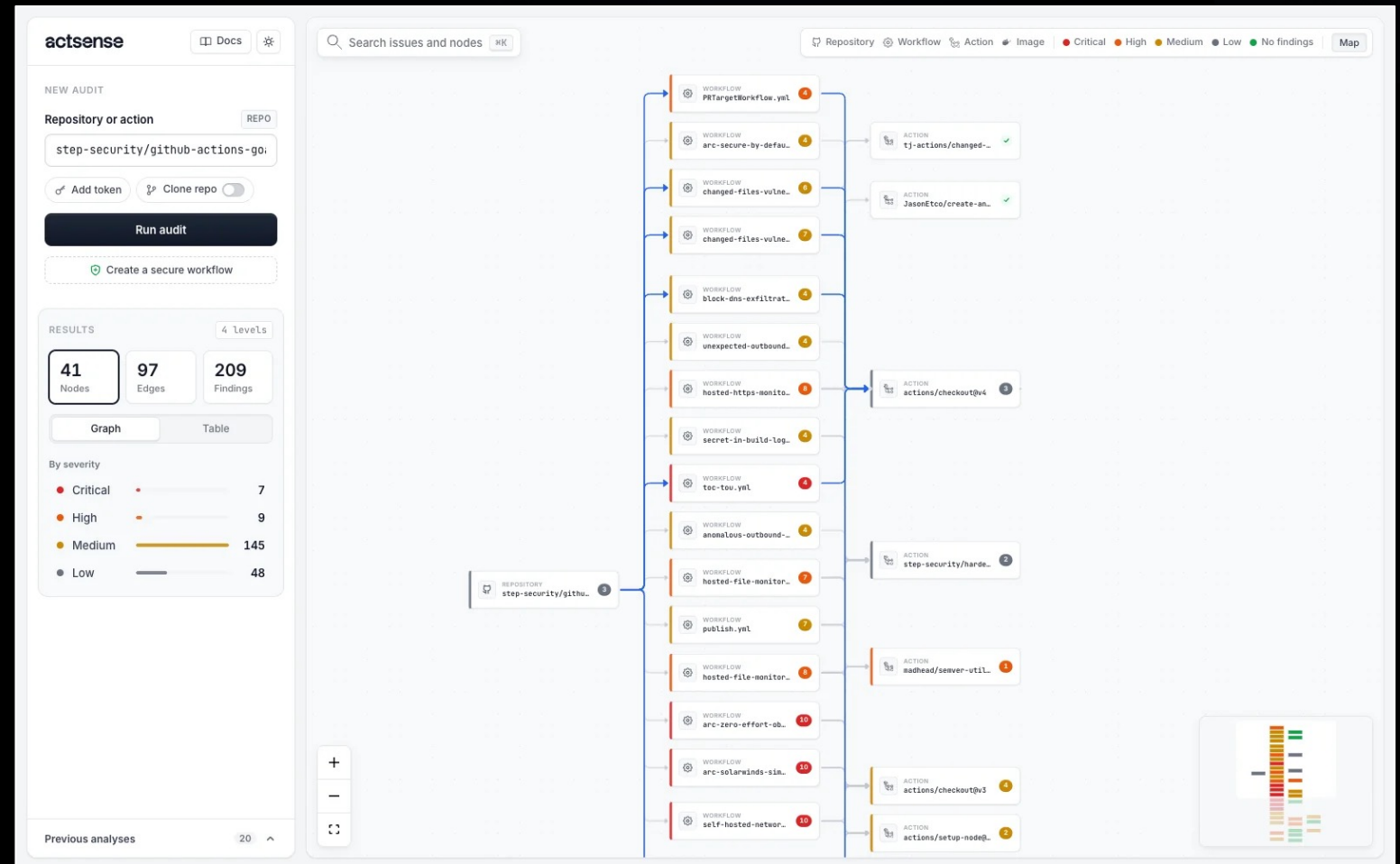
They open a PR. Your CI runs it - with your secrets.

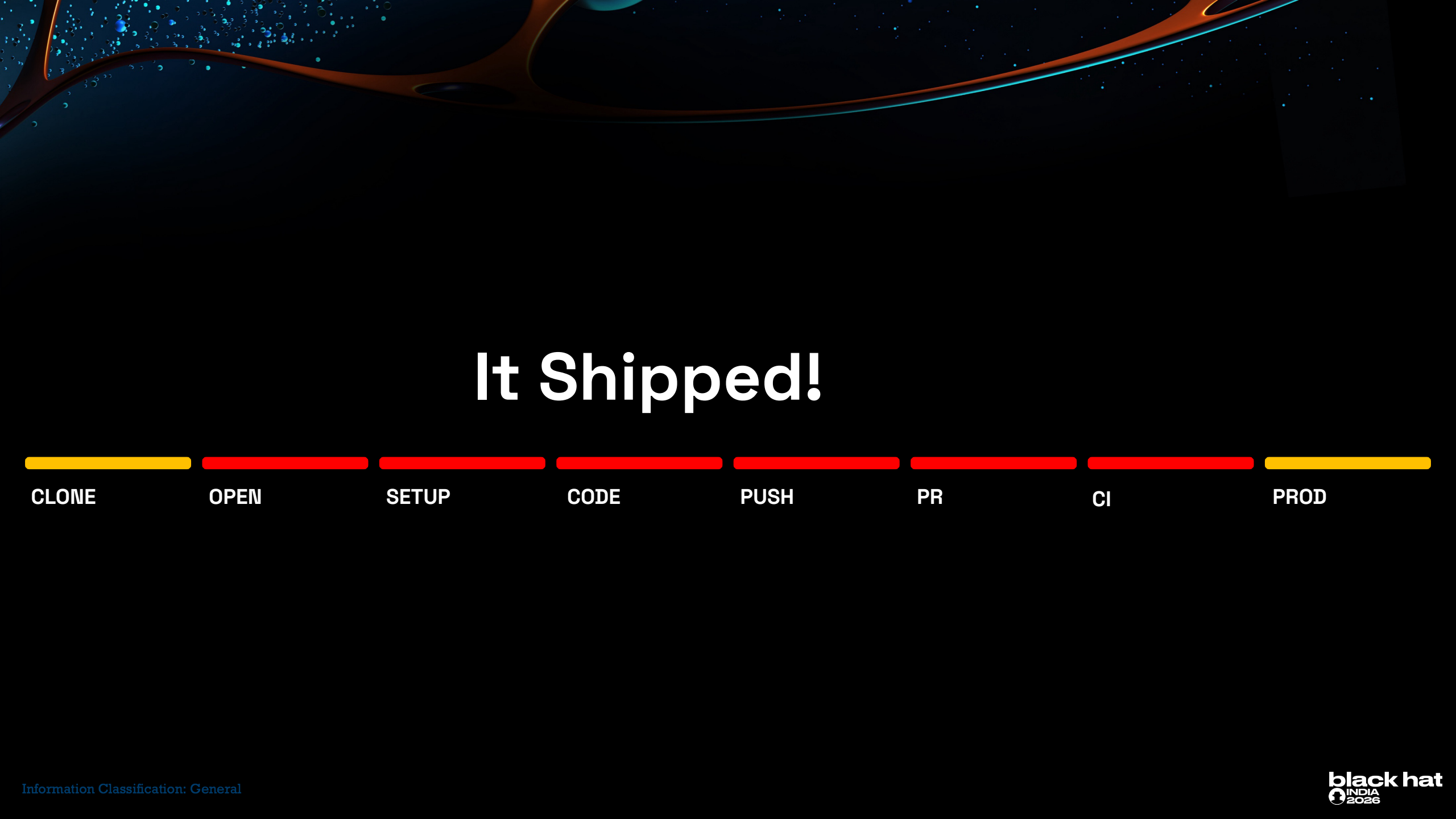
A fork's code, its branch name, its PR title - all become input your pipeline executes, holding a write token and your keys.

<code>pull_request_target</code>	→ runs fork code WITH secrets
<code>\${{ pr.title }}</code> → run:	→ their text becomes your command
<code>\$GITHUB_ENV / \$PATH</code>	→ their input rewrites later steps
self-hosted runner	→ their job lands on your box

Presenting actsense.dev

open-source workflow auditor
self-hosted

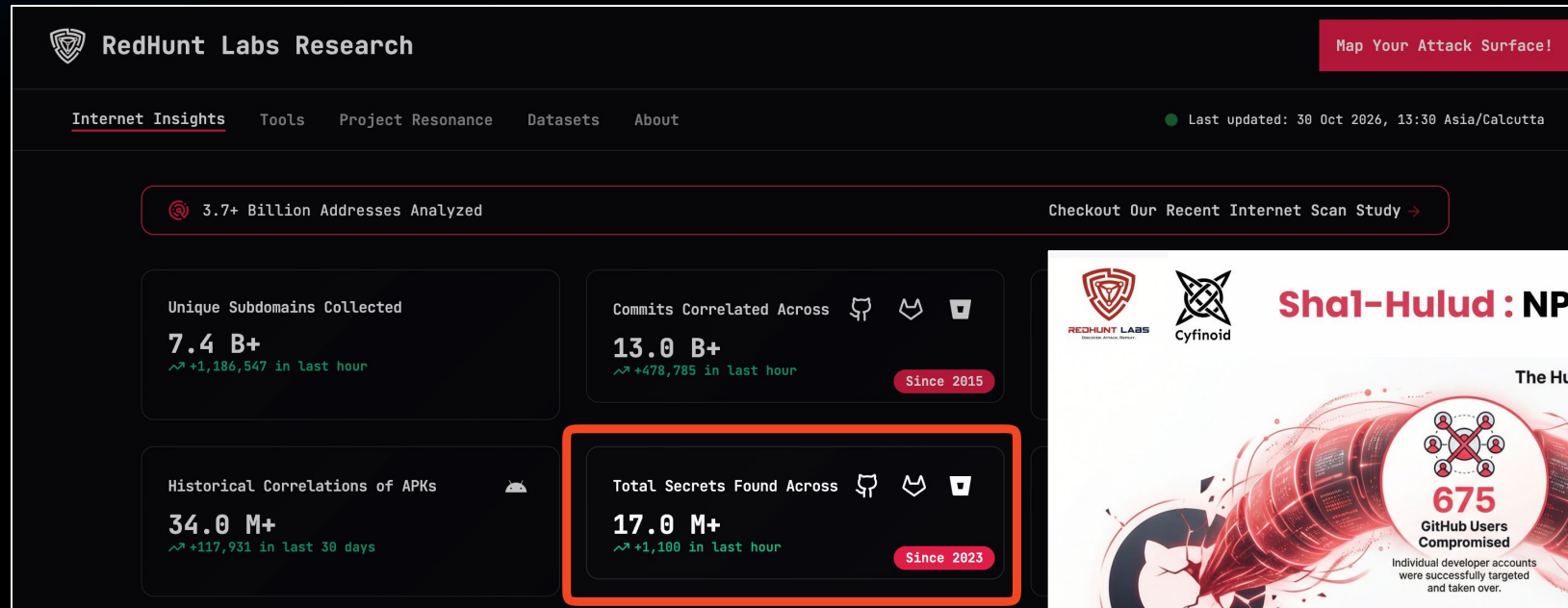




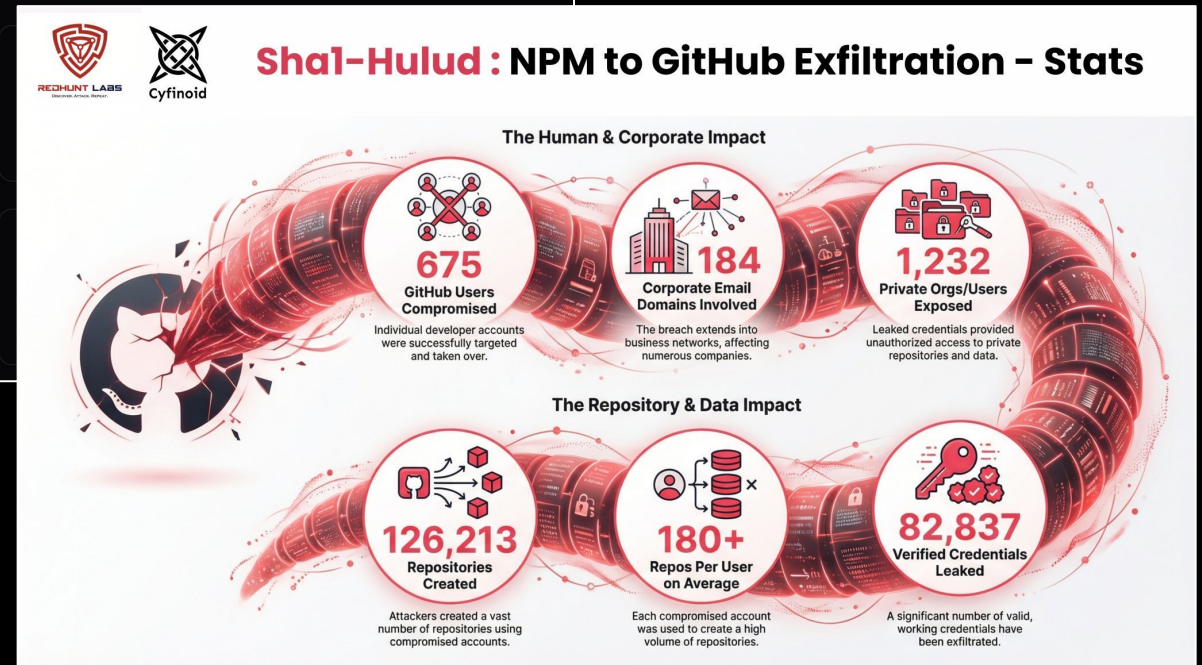
It Shipped!



Initial Access, how?



<https://kumarashwin.com/events/the-breach-is-over-the-exposure-is-not-defcon-recon-village-2026>



Final note!

- Not one of these is a bug. Each is a feature.
- The repo is input. The agent is input. The PR title is input. And input runs.
- **autorun.inf** → **.pth** → **AGENTS.md** — same primitive, thirty years apart.
- We keep inventing new ways to run a stranger's code on open.
- The attack surface isn't your code. It's everything that runs to get to your code.

Tools & Takeaways!

 **lote.guide**

lote.guide



actsense.dev



cot.run

Thanks!

Kumar Ashwin (@OxCardinal)



linkedin.com/in/OxCardinal