The REA Setup Playbook

The full install, the exact commands, and the fix table for Reverse Engineer Anything, the open-source MCP server that lets your AI agent study any app feature and rebuild it.

Why I'm sending you this

A tool called REA, short for Reverse Engineer Anything, hit number one on GitHub trending. It's an MCP server that connects Claude Code and other AI agents to real analysis tools, so your agent can study how a feature actually works even when you have zero access to the source code. Pointing an agent at a problem and saying 'figure this out' usually gets you a guess. REA gives the agent something to trace instead of guess.

Everything in this guide comes from the project's own README and docs. No code or command here is made up.

01

How it actually works

1. REA registers itself as an MCP server with your coding agent, then installs workflow instructions that teach the agent how to use its analysis tools.
2. When you point the agent at a feature, REA inspects it directly: the compiled binary, the JavaScript bundle, the network traffic, or whatever format applies, all running locally on your machine.
3. The agent traces the behavior down to the actual code, generates pseudocode or assembly where needed, and hands back findings with evidence attached instead of a surface-level description.

02

The one thing almost everyone gets wrong

People assume REA needs the decompilers (IDA, Hopper, Ghidra) installed before it does anything useful. It doesn't.

> **WATCH OUT**
>
> Decompilers are listed as optional in the README, and they only matter for native binary work (compiled C/C++ apps, for example). For JavaScript apps, Electron apps, and websites, which cover most SaaS products and browser extensions, REA's own built-in analyzer does the whole trace with nothing extra installed. Don't block your first run on installing IDA or Ghidra if you're studying a web app.

03

The commands and the agent prompt

SOURCE CODE

The full REA repository:

https://github.com/morluto/rea

SETUP

```
npx rea-agents setup
```

GLOBAL INSTALL (OPTIONAL)

```
npm install --global rea-agents
```

UPDATE

```
rea update
```

ANALYZE A JAVASCRIPT OR ELECTRON APP

```
npx -y rea-agents@latest analyze-javascript-application /absolute/path/to/app --json
```

SITES SHOWN IN THE VIDEO

- REA on GitHub (morluto/rea)
  https://github.com/morluto/rea

Set it up in Claude Code or Codex

1. Open a terminal in any project folder. REA doesn't need its own dedicated project, you run it from wherever your agent session already lives.
2. Clone the repo if you want to read the source yourself: git clone https://github.com/morluto/rea. This step is optional, the setup command below works without cloning anything.
3. Run npx rea-agents setup. This detects your agent (Claude Code, Codex, Cursor, Gemini CLI, or Grok Build) and registers REA as an MCP server automatically, no manual config file editing.
4. Check your Node version first if setup fails: REA needs Node.js 22.x (22.19 or later), 24.x (24.11 or later), or 26+. Run node --version to confirm, and update Node if you're on an older release.
5. On Linux, make sure you have a procps-compatible ps command available, REA uses it for process checks during analysis.
6. Start a new terminal window after setup finishes, so your agent picks up the newly registered MCP server.
7. Give your agent a real target: point it at an app feature you want to understand, using a prompt like the one in the next section.

💡 **QUICK TIPS**

- If you're not sure which agent you're running, just run the setup command, it detects Claude Code, Codex, Cursor, Gemini CLI, and Grok Build on its own.
- You don't need API keys or an account to use REA itself, it's a local MCP server, not a hosted service.

04

The prompts I actually run

STUDY A FEATURE WITH NO SOURCE CODE

```
Understand how [FEATURE NAME] works in [APP NAME], show me the evidence, and build a similar feature for my project.
```

RUN THE RAW ANALYZER ON A JAVASCRIPT OR ELECTRON APP

```
Run REA's analyze-javascript-application tool against [ABSOLUTE PATH TO APP] with --json and summarize the output for me.
```

ASK FOR A REBUILD PLAN BEFORE ANY CODE GETS WRITTEN

```
Before you write any code, lay out the trace you'd follow to understand [FEATURE NAME] in [APP NAME], and tell me what evidence you'd expect to find at each step.
```

CONFIRM THE SETUP WORKED

```
Check that the REA MCP server is registered and list the analysis tools it has access to.
```

When it breaks

ERROR	FIX
Setup command fails or hangs	Check your Node.js version with node --version. REA needs 22.19+, 24.11+, or 26+. Update Node and rerun npx rea-agents setup.
Agent doesn't see REA as an available tool	Open a brand new terminal window after running setup. MCP servers are registered at session start, an already-open agent session won't pick up the new connection.
analyze-javascript-application fails on a path	Use an absolute path, not a relative one. Something like /Users/you/Downloads/app, not ./app.
Native binary analysis comes back incomplete	That's expected without a decompiler installed. Install Hopper, Ghidra, or IDA if you're specifically working with compiled native binaries, not JavaScript or web apps.
rea update doesn't find the command	You installed with npx instead of a global install. Run npx rea-agents@latest setup again to pull the latest version, or install globally first with npm install --global rea-agents.
Linux process checks fail silently	Confirm you have a procps-compatible ps command available on your system, REA's process checks depend on it.

Ready to go deeper?

This playbook is the short version. Inside the hub you get the full build walkthroughs, the prompt library, and the systems I use to ship this in a weekend.

Join the Build →

https://hub.digicuratoragency.com/join