SCI-FI LABS

A FIVE-PART SERIES

Jailbreak Your Laptop

Own the OS. Own the apps. Own the stack.

PAUL VISCIANO

paulvisciano.com

HOW-TO GUIDE · SCI-FI LABS

Jailbreak Your Laptop

Own the OS. Own the apps. Own the stack. A five-part recipe for a computer that is actually yours.

To see why this matters, go back to where jailbreaking began. Back then the lock was physical — a phone locked to one carrier, sealed by firmware, and the only way out was to crack the firmware itself. In the summer of 2007, a seventeen-year-old named George Hotz — geohot — spent five hundred hours soldering and coding to unlock the original iPhone. He posted the video from his parents' kitchen: "This is the world's first unlocked iPhone." It was breaking news. The New York Times, CBS, CNBC, Wired — the story spread across the world in a weekend. He traded one of the unlocked phones for a Nissan 350Z. He was still a teenager.

Today the lock isn't the phone. It's the operating system on the laptop you use every day — the vendor's assumptions baked into the software, deciding what runs, what phones home, what you can change. Jailbreaking your laptop is the same idea: crack the lock the vendor shipped, on whatever machine you have. Mac, PC, or a Linux box you already run.

The payoff is transparency and editability. On a locked OS you can still inspect processes — the difference is what happens next. On Omarchy you get a curated, minimal set of services you chose, no vendor telemetry baked in by default, and the ability to override everything through AI. Something not working? Describe it and the local model rewrites the config. Don't like how it looks? Ask for a custom theme. A game that won't launch, a driver that misbehaves — those become easily tweakable through conversation, with every change visible and reversible. The lock you cracked becomes a machine you can talk to.

This is the roadmap to sovereignty. Every layer already works. Follow the steps and you end up with a computer that is truly yours.

The Ingredients

A laptop — 16 GB works. No GPU required. Mac, PC, or Linux.

Omarchy — the owned OS. omarchy.org · iso.omarchy.org · github.com/omacom/omarchy

Bonsai — personal AI brain (1-bit, ~1.15 GB for 8B). prismml.com · huggingface.co/prism-ml/bonsai

llama.cpp — runtime for Bonsai. Ships with its own UI. github.com/ggml-org/llama.cpp

Open WebUI — daily workspace. github.com/open-webui/open-webui · Graph fork: github.com/paulvisciano/open-webui

Stronger models — Grok 4.6 via OpenCode, or GLM 5.2 / 5.1, for Let AI Cook.

Vercel + GitHub + a domain — the public layer only.

> **Proof it works.** I already have the whole stack running. Open WebUI with the Knowledge Graph. Bonsai served through llama.cpp. The portfolio live on paulvisciano.com. Spatial apps — Where is Paul? and Musical Cubes — built and tested on this machine. The steps below are what I actually did.

> **Two layers of AI.** Bonsai is the personal layer — journaling, talking to a local model, the knowledge graph. Grok 4.6 and GLM 5.x are for the moments you need more power: setting up software, coding, multi-step installs. Reach for those when building. Hand daily use back to Bonsai.

Part 1 — PWN the OS

Crack the lock the vendor shipped. Replace their assumptions with an OS you control. Same defiance as geohot — the lock just moved from firmware to the system sitting on the machine.

DO THIS · OMARCHY.ORG/MANUAL/GETTING-STARTED

01 **Download the ISO**
omarchy.org or iso.omarchy.org

02 **Flash it to a USB stick**
balenaEtcher on Mac/Windows, caligula on Linux

03 **Boot the stick and open BIOS**
Turn off Secure Boot and TPM. Use a wired or 2.4 GHz keyboard — encryption will not accept Bluetooth at startup.

04 **Run the configurator**
Full-disk wipes the drive — back up first. Free-space dual-boots; turn off BitLocker in Windows first. Encryption is on by default.

05 **Reboot**
Under a minute on fast machines, no more than five on older ones. The services running are the ones you chose.

Result. The machine is yours. No cloud middleman. No telemetry you can't refuse. You get a curated, minimal set of services you chose — and after this step, you can override themes, drivers, and game fixes through conversation, with every change visible and reversible.

> **This step is manual.** The OS install is the one part you do by hand — no AI in the loop. You want to feel the lock crack yourself. AI comes in from Part 2 onward, once the machine is yours.

Part 2 — Run Your Local AI

Three surfaces, one machine: llama.cpp's own UI, Open WebUI as the daily workspace, and Graph as memory. Do this by hand, or paste the prompt into an agent. Pick one path. Don't run both.

DO THIS BY HAND

01 **Get llama.cpp running**
github.com/ggml-org/llama.cpp — it ships with its own UI

02 **Pull Bonsai**
huggingface.co/prism-ml/bonsai — start with Bonsai-8B-Q1_0.gguf. Serve it: ./llama-server -m Bonsai-8B-Q1_0.gguf -c 0 --port 8080

03 **Point Open WebUI at localhost:8080**
github.com/open-webui/open-webui — chats, notes, search

04 **Want the Knowledge Graph?**
Use the fork: github.com/paulvisciano/open-webui. Graph lives under the Graph menu — chats as cards on a canvas, no cloud

05 **Bind to your LAN IP**
Any device on the Wi-Fi — phone, laptop, tablet — hits the same assistant

Result. Your assistant and your memory live on your machine. Phone and laptop share them.

OR LET AI COOK

Prefer not to run the steps yourself? Paste this into OpenCode + Grok 4.6, Claude Code, Hermes, or OpenClaw. Then hand daily use back to Bonsai.

```
Set up llama.cpp on my laptop, pull the Bonsai 8B Q1_0 GGUF, serve it on port 8080,
install Open WebUI and point it at that endpoint, then clone
github.com/paulvisciano/open-webui for the Knowledge Graph under Graph. Bind everything
to my LAN IP so any device on the Wi-Fi can reach it. No cloud, no API keys.
```

Part 3 — Create Your Own Apps

Where is Paul? saves the journey — pins, comics, blogs, photos, video.
Musical Cubes is an original: tracks as rotating cubes. Both run on localhost.
Yours will look different.

DO THIS BY HAND

01 **Clone or start from a template**
Fork an open-source app you like, or begin from scratch

02 **Point it at localhost:8080**
No hosted API. No usage meter.

03 **Stop forking. Scaffold an original**
Build something that talks to your local model

04 **Run it on the machine from Part 1**
No account. No subscription. No telemetry you can't refuse.

Result. Apps stop being products someone sells you. They become infrastructure you run.

OR LET AI COOK

Prefer the agent path? Paste this into OpenCode + Grok 4.6, Claude Code, Hermes, or OpenClaw.

```
I have a forked web app that currently calls OpenAI's API. Rewrite it to call a local
llama.cpp server on http://localhost:8080 instead. Then scaffold my first original app
on this Omarchy machine that talks to the same local model. No cloud, no API keys.
```

Part 4 — Create Your Own Site (The Public Layer)

This is the public face — not the brain. Claim a domain. GitHub is source of truth. Vercel deploys. The internet gets a site you chose. Proof: paulvisciano.com — work and blogs on a domain I host.

DO THIS BY HAND

01 **Claim a domain**
Namecheap, Cloudflare, Porkbun — whatever registrar you trust

02 **Push a repo to GitHub**
This becomes the source of truth

03 **Import the repo into Vercel**
It deploys on every push to main

04 **Wire the domain in**
Vercel → Settings → Domains. Set the A record or CNAME at the registrar.

Result. A public face on the internet that no platform can take away.

OR LET AI COOK

Prefer the agent path? Paste this into OpenCode + Grok 4.6, Claude Code, Hermes, or OpenClaw.

```
Take me from zero to a live personal site: buy a domain, create a GitHub repo, push a static site, connect Vercel so every push to main deploys, add the custom domain, and give me the exact DNS records to set at my registrar. Verify with curl that it returns 200.
```

Part 5 — Publish If You Want. Reclaim Your Data.

Publishing is optional. The archive is not. Publish only what you choose. Then pull your data home.

DO THIS BY HAND

01 **Publish only if you choose to**
Push an app to the repo from Part 4. If you don't want it public, skip this.

02 **Export your data**
Google Photos, WhatsApp, Instagram, Facebook, bank statements

03 **Import the archive onto this machine**
The copy that lives here is the one you own

04 **Drop it into the stack**
Memory → Open WebUI Graph. Journey → Where is Paul?. Work → the portfolio.

Result. You own the OS, the AI, the apps, the site, and the archive. Private. Secure. Sovereign.

OR LET AI COOK

Prefer the agent path? Paste this into OpenCode + Grok 4.6, Claude Code, Hermes, or OpenClaw. Keep the import local.

```
Optionally publish my original app to my custom domain via Vercel on git push. Then walk me through reclaiming my data from Google Photos, WhatsApp, Instagram, Facebook, and bank statements. Store the exports on this machine and import them into the knowledge-graph app talking to Bonsai on localhost:8080. Keep the archive fully local.
```

The finished dish. You reclaimed the computer. You own the OS. You run your own AI. You run your own apps. You have a public layer that is yours. Your data is home. The road to sovereignty isn't a destination — it's a recipe you can run again, on any machine, whenever someone tries to take the lock back.