

Episode AI: The Phantom Dependency Menace

What's in your AI Code?

ENDOR LABS

Darren Meyer, Staff Research Engineer — 26. June 2024

Why us?

- Focus on dependency management
 - OSS libraries
 - LLM Models
 - Other stuff (CI actions, etc.)
- Strong commitment to accuracy and completeness
- Context and program analysis baked in from day one

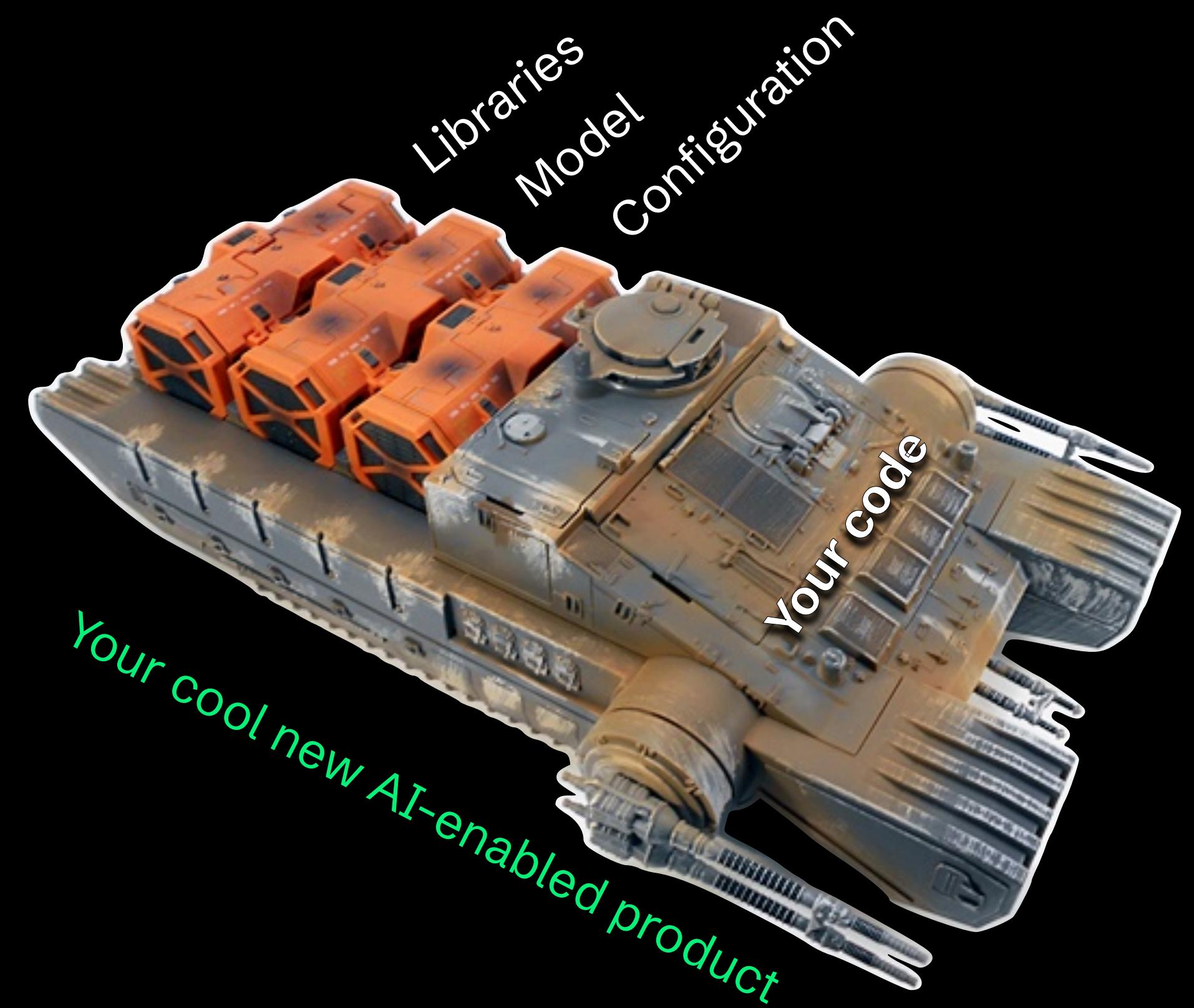


Why me?

- 20 years in AppSec
 - Built and ran security programs
 - Practical researcher for over 9 years
- Software engineering background
- General nerd



What do we mean by “AI Code”?



How dependencies work

Package Managers and Runtimes tend to operate completely decoupled, like ships passing in the night

1. Developer starts a new project that uses a couple of dependencies
2. Developer creates a **manifest** file (requirements.txt, package.json, pom.xml, etc.) to declare **direct** dependencies
3. Build system runs **package manager** and the direct dependencies bring along several other **transitive** dependencies
4. Package manager copies the files in a directory
5. **Runtime/compiler** loads dependencies as needed during execution



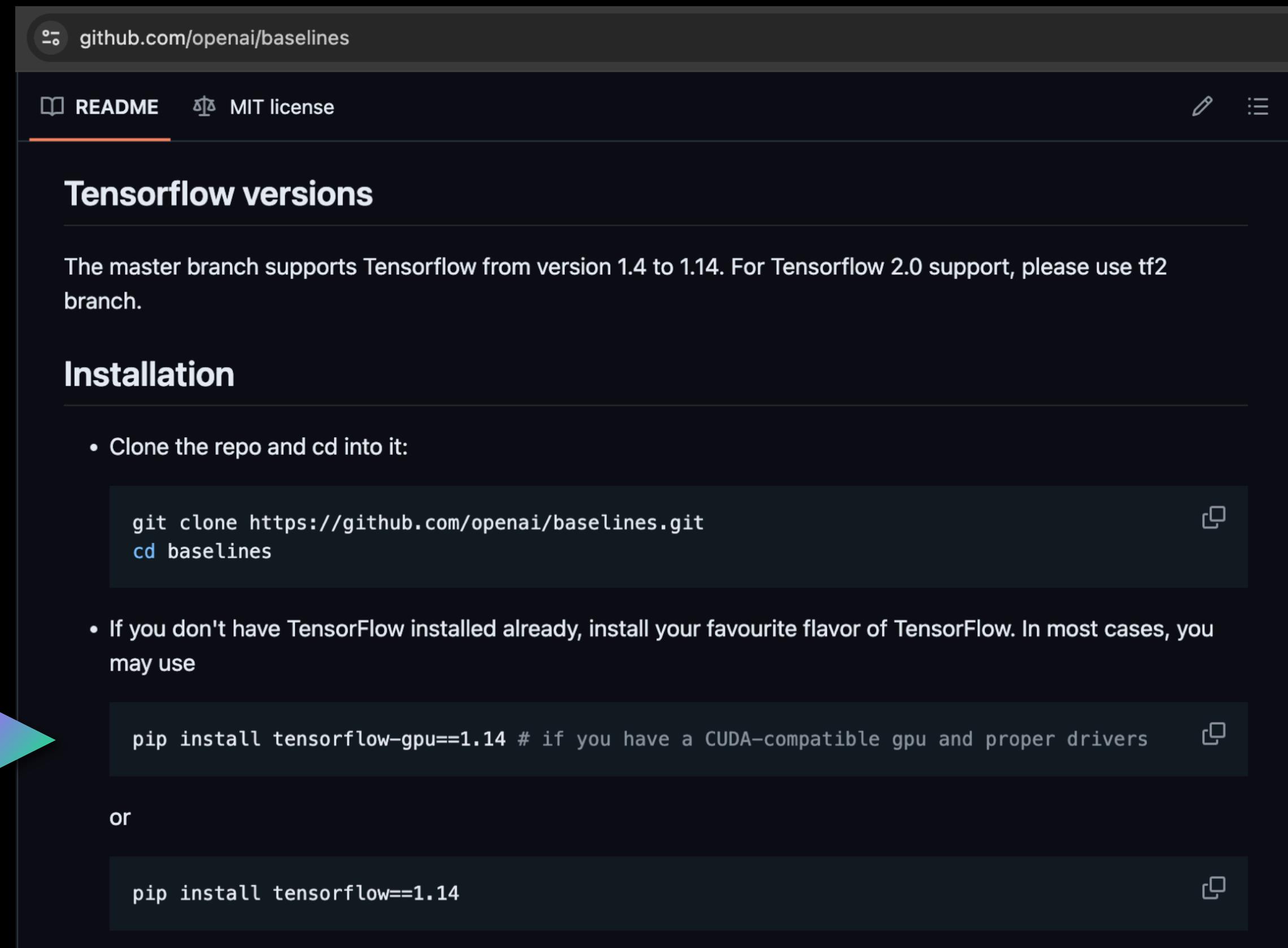


Things get out of sync

It is unavoidable

- Developers import **new dependencies** without updating the manifest file (possible in python, Javascript, scripts etc)
- In some cases dependencies are there **in the environment** (like global python or node packages)
- In some cases dependencies are **for testing/dev** (example: storybook in Javascript)
- In some cases dependencies are **removed from the code** but not from the package manager manifests

Your manifest can lie



The screenshot shows the GitHub README page for the `openai/baselines` repository. It includes sections for `Tensorflow versions` (supporting 1.4 to 1.14), `Installation`, and command-line installation instructions:

- Clone the repo and cd into it:

```
git clone https://github.com/openai/baselines.git  
cd baselines
```
- If you don't have TensorFlow installed already, install your favourite flavor of TensorFlow. In most cases, you may use:

```
pip install tensorflow-gpu==1.14 # if you have a CUDA-compatible gpu and proper drivers
```

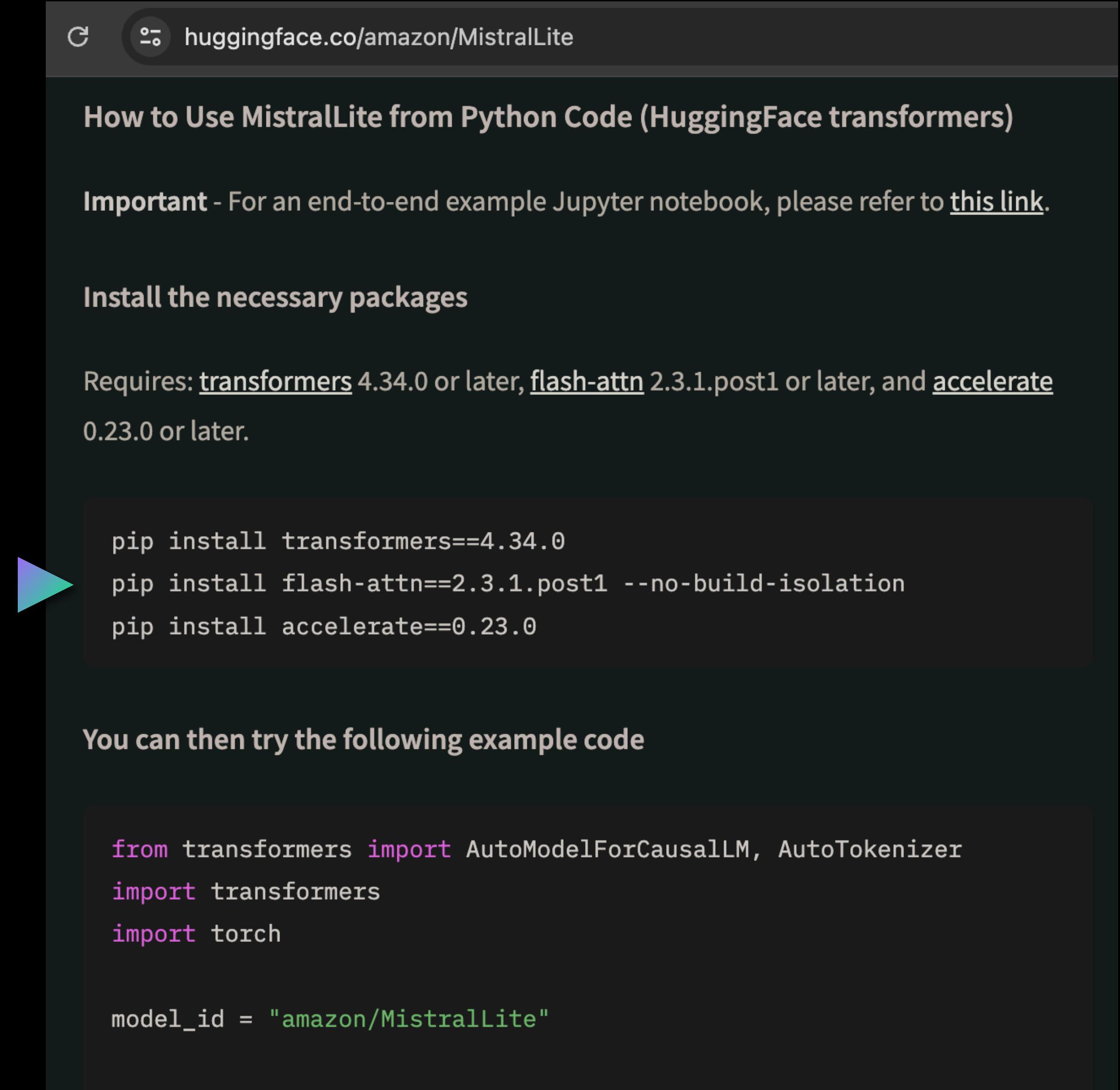
 or

```
pip install tensorflow==1.14
```

OpenAI's Baselines library

- “Just `pip install` a dep”
- Baselines *won’t function without the right version*
- You’ll never see it in a manifest or lock file
- SCA / dep tree tools (usually) won’t see it

Models suggest this pattern often



The screenshot shows a browser window with the URL huggingface.co/amazon/MistralLite. The page title is "How to Use MistralLite from Python Code (HuggingFace transformers)". A note says "Important - For an end-to-end example Jupyter notebook, please refer to [this link](#)". A section titled "Install the necessary packages" lists requirements: "Requires: `transformers` 4.34.0 or later, `flash-attn` 2.3.1.post1 or later, and `accelerate` 0.23.0 or later." Below this is a code block with a play icon containing three pip install commands:

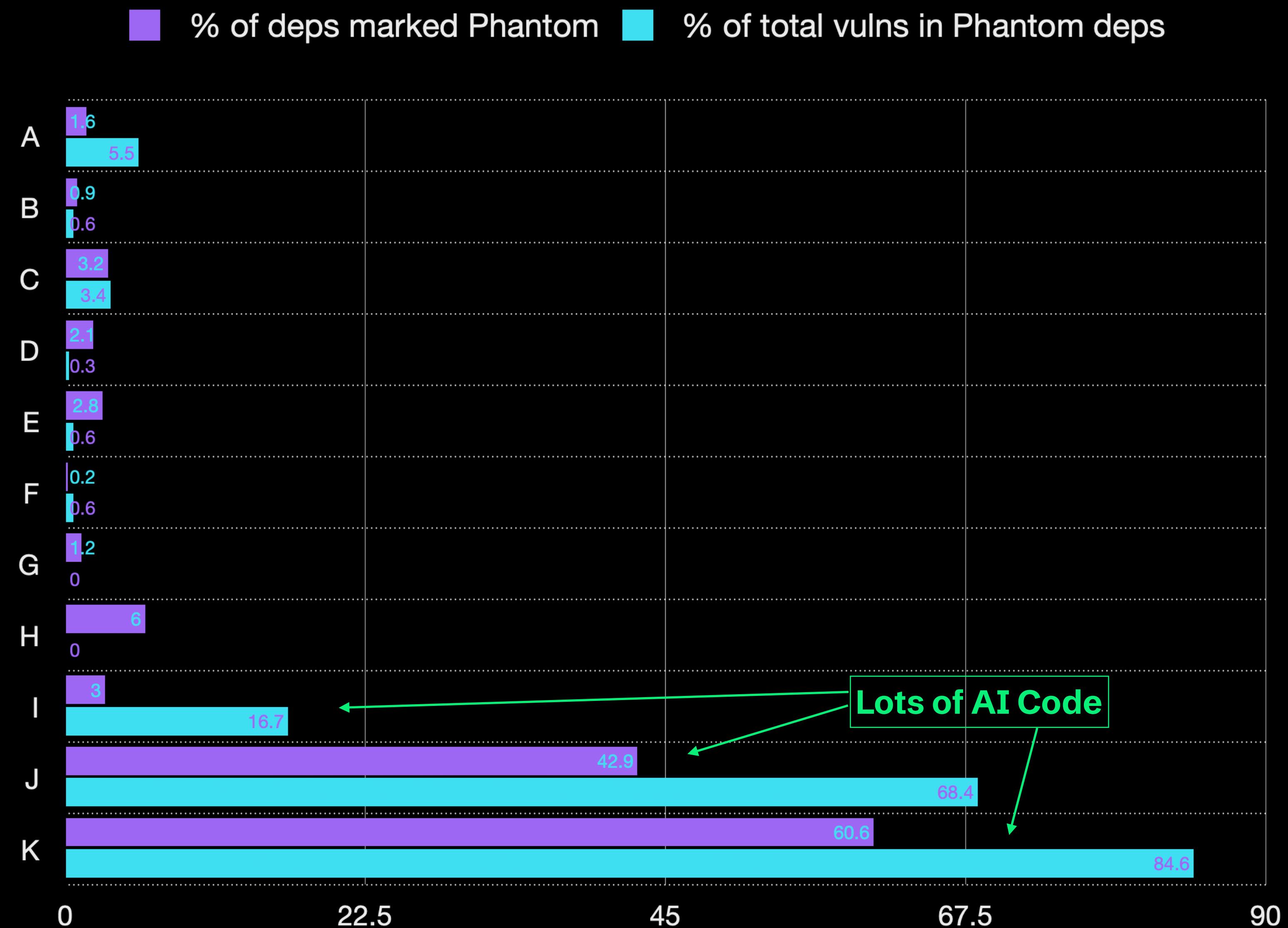
```
pip install transformers==4.34.0  
pip install flash-attn==2.3.1.post1 --no-build-isolation  
pip install accelerate==0.23.0
```

A section titled "You can then try the following example code" contains the following Python code:

```
from transformers import AutoModelForCausalLM, AutoTokenizer  
import transformers  
import torch  
  
model_id = "amazon/MistralLite"
```

The Phantom Dependency Menace

- Dependencies that are either “provided” by the system are assumed to be downloaded manually
- Scripts, containers, and so on
- Often depend on the target platform
- Dependencies that are required for building an application that are not supposed to be used at runtime but are actually used
- Very common in NPM: see storybook for example

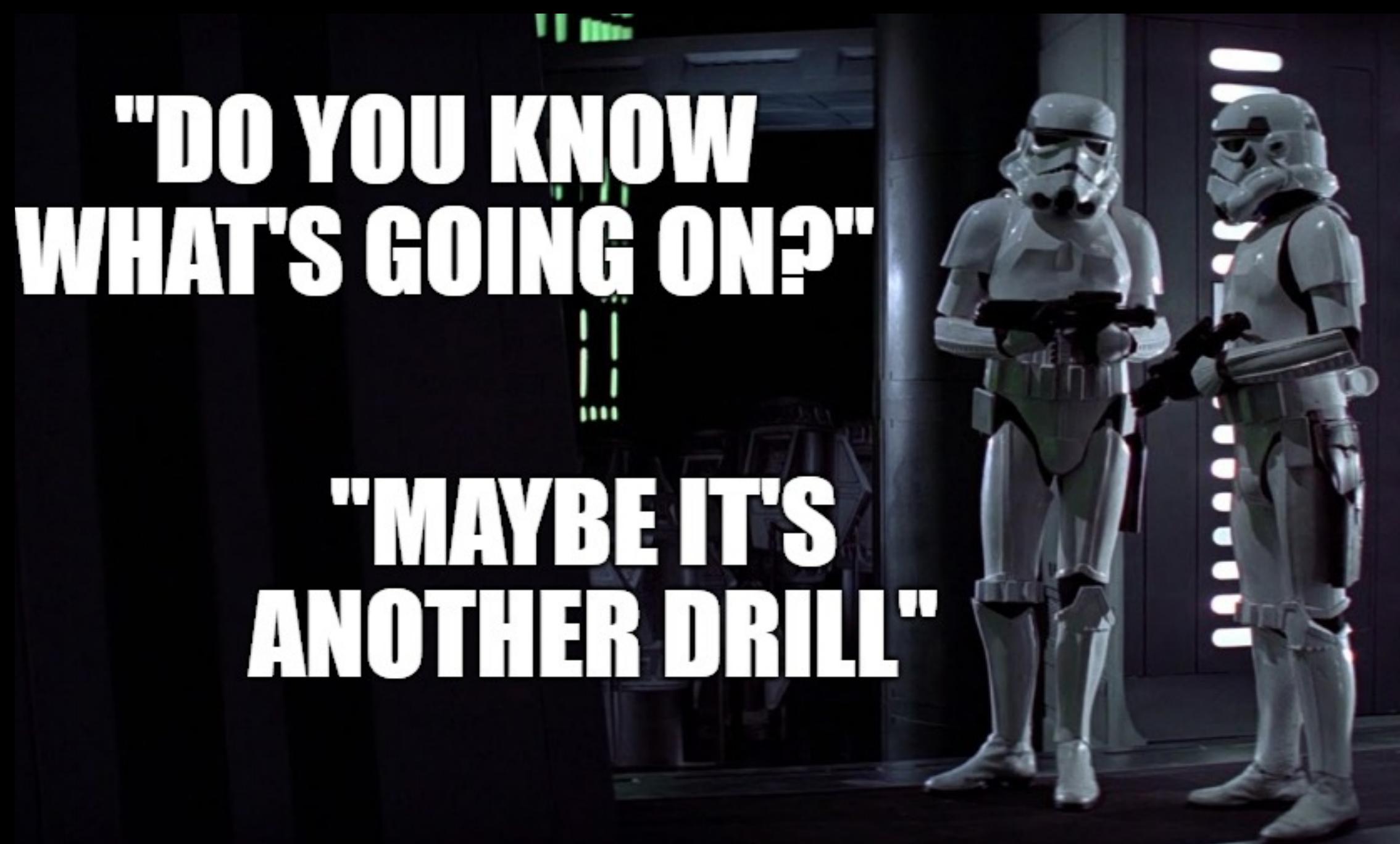




Security Challenges

- False sense of security — tools can't see what's not in a manifest, so you miss risks that might be relevant
- Inaccurate compliance data — your SBOMs aren't reporting everything in use. Auditors are unhappy if they catch you
- Dev / prod differences — can't rely on the version I see in dev pipelines being the same thing that's in production

Why are tools blind?



Many tools trust the manifest or lock files, and don't account for the ways those can lie

1. Phantom Dependencies (false negatives)
 - Brought by the system, runtime or other scripts
2. Mis-used dependencies (false negatives)
 - Dependencies brought as “test/dev” used in runtime
3. Direct use of transitives (unreliable fixing)
 - Dependencies brought in as transitives and used directly without knowledge
4. Unused dependencies (false positives and noise)
 - Dependencies brought in the manifest but not used by the code

Program Analysis FTW

What matters is which packages the code **actually uses**

1. Source of truth is actually **the source code**
 - a. Analyze the code
 - b. Create an Abstract Syntax Tree
 - c. Analyze types and call flows
 - d. Create a **call graph**
2. Correlate the **dependencies used by the code** with the dependencies fetch by the package manager or available in the file system
3. Create a unified view



Example: Python

Use the source

1. Import dependency ► Call graph ► “Is it used?”
 - a. Repeat for all it’s dependencies ([transitive](#))
2. Compare [dependency graph](#) with versions installed on system and defined in manifest
3. [Correlate](#) and unify results
 - a. Makes accurate [SBOM](#) and [VEX](#) possible



The Shameless Pitch

All Projects

openai/baselines · [master](#)

- Accurately identify all dependencies in use, even if they're “phantom”
- Provide clear mapping and pathway data
 - What uses it?
 - Directly vs. Transitively?
- Find out when transitive deps are being directly used
 - Avoid the noise to devs by knowing whether a risk is actually along a call path

OVERVIEW FINDINGS 38 PACKAGES 1 DEPENDENCIES 36 TOOLS 0 PR RUN >

Packages 36 CI Workflows BETA

Ecosystem ▾ ×

Showing 1 of 36 dependencies

Dependency	Requested Version	Type	Dependent Packages	Reachable	Visibility
tensorflow @2.16.1	N/A	Direct	1	✓	⊕
Phantom					

tensorflow 2.16.1

[View Details](#)

OVERVIEW SCORES

FINDINGS	C 0 H 0 M 0 L 0
DEPENDENCIES	0
LICENSES	N/A
PHANTOM DEPENDENCY ⓘ	Yes
RESOLUTION TIME	Jun 22, 2024

Dependency Path

```

graph TD
    A[openai/baselines] --> B[pypi://baselines@0.1.6]
    B --> C[pypi://tensorflow@2.16.1]
  
```

< 1 of 1 paths >

Dependency Specification

DEPENDENT PACKAGE	baselines
RESOLVED VERSION	2.16.1



Sign up for a Free Trial of Endor Labs

<https://www.endorlabs.com/free-trial>

The diagram features a central dark oval containing a white 'E' logo with a green dot. Five orange warning icons with exclamation marks are connected by lines to the oval from the left. From the right side of the oval, three green lines extend to callout boxes: '3 Valid Secrets' at the top, 'Reachable CVEs' in the middle, and '121 Insecure Repos' at the bottom. Below the diagram, large white text reads 'Secure open source ~~open source~~ everything your code depends on'. At the bottom, smaller text states 'Identify, prioritize, and remediate SDLC risks without slowing down developers.'

Secure ~~open source~~ everything
your code depends on

Identify, prioritize, and remediate SDLC risks without slowing
down developers.

Greetz and Thanks

Dimitri Stiladis, Henrik Plate
original research, program analysis design

Jamie Scott
models, reviews

Pia Rodil
Food and logistics support for this meeting

OWASP Membership and OWASP MSP
None of this is possible without the commitment of members and leaders at OWASP

Many, many more...