# Who's reproducing the reproducible images?

FOSDEM 2026

Alex Pyrgiotis (@apyrgio)

FREEDOM OF THE PRESS FOUNDATION

# What we'll talk about

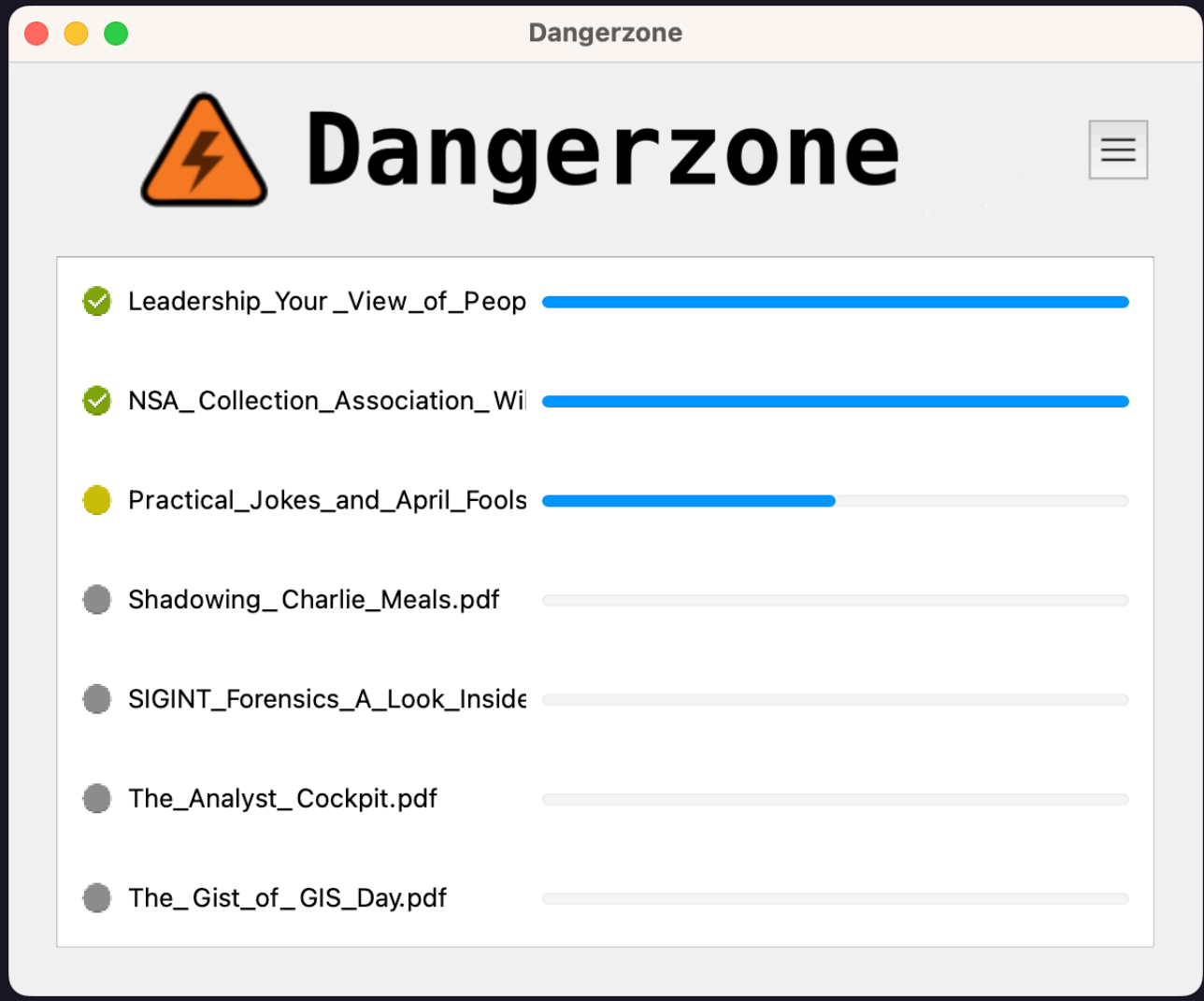
- Why on earth should I care about reproducible images?
- How do I build one? (+ some new tools)
- What else do I need?


Trust issues

# Dangerzone

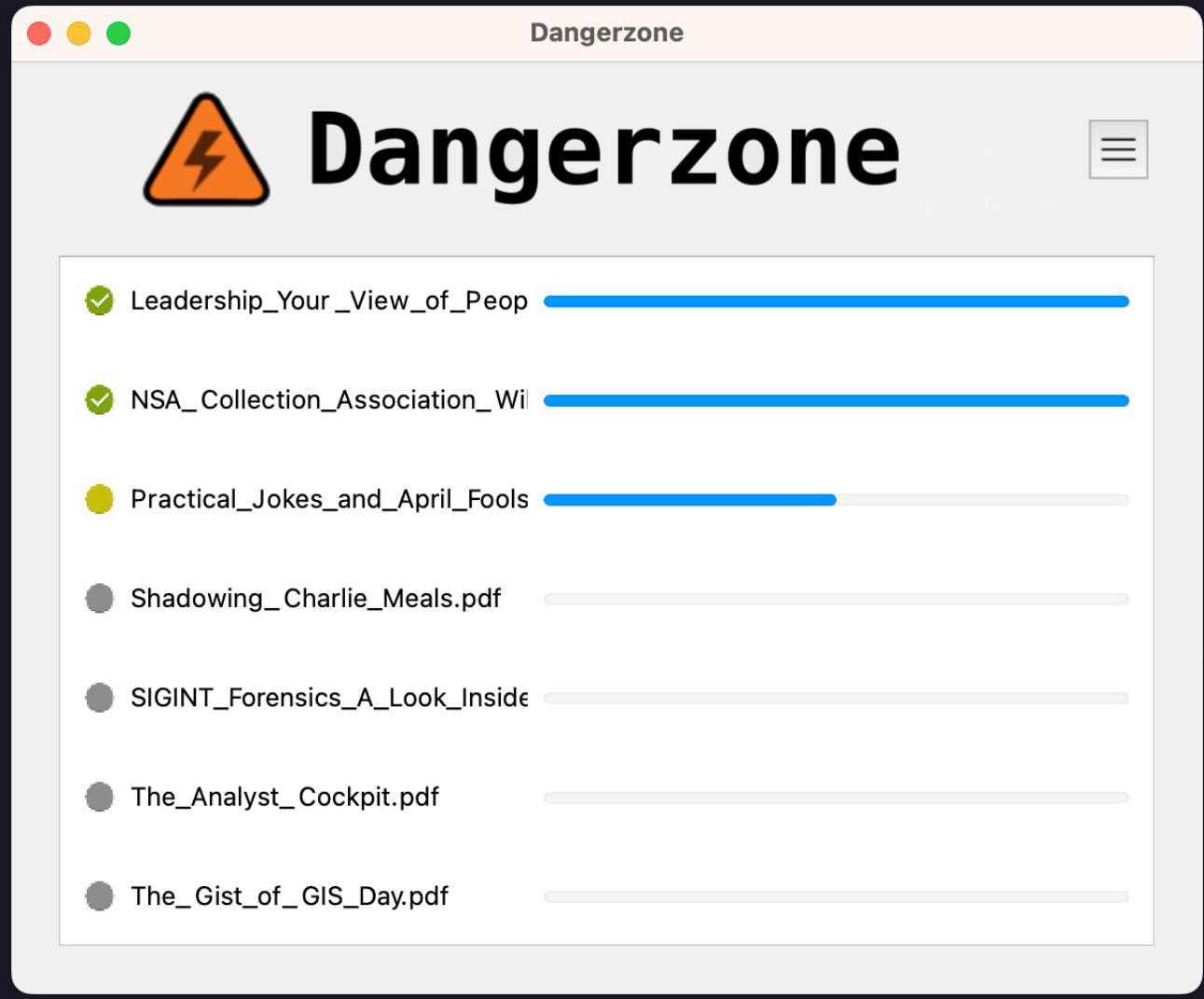
- Desktop application that uses containers to sanitize untrusted documents
- Auto-updates its container image
- Used by journalists and activists

Our threat model is a bit heightened.



# Dangerzone

1. We build our images nightly via GitHub Actions
2. We can't just trust they were built correctly
3. Even if we did, our users should be able to verify



# Why reproducibility matters to us

1. **Auditability**: Anyone can check that the source code matches the image
2. **Detection**: Backdoors introduced during build (in CI or build nodes) become visible.
3. **Resilience**: Our laptops, hardware keys, and CI pipelines are no longer a viable target

That's why our container image is **bit-for-bit reproducible**.

# But what is a "reproducible build"?

> *"A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts."*
>
> — *Reproducible Builds project*

# Let's create a reproducible image!

# Container reproducibility 101

Docker (Buildkit) and Podman (Buildah) have built-in support:

1. `SOURCE_DATE_EPOCH`: Forces a fixed timestamp for file metadata.
2. **Rewrite Timestamps**: Ensures layer tarballs use that fixed epoch.

# Container reproducibility 101

## Buildkit (Docker)

```
export SOURCE_DATE_EPOCH=...
docker buildx build --output type=docker,rewrite-timestamp=true .
```

## Buildah (Podman)

```
podman build --source-date-epoch ... --rewrite-timestamp .
```

# Is this image reproducible?

```
FROM debian:bookworm-20230904-slim
RUN apt-get update && apt-get install -y gcc
```

# Is this image reproducible?

```
$ export SOURCE_DATE_EPOCH=1677619260
$ docker buildx build --no-cache \
    --output type=docker,dest=img1.tar,rewrite-timestamp=true .
# sha256:81ebf01e...
```

One more check:

```
$ docker buildx build --no-cache \
    --output type=docker,dest=img2.tar,rewrite-timestamp=true .
# sha256:231a1e36...
```

Oh, the digests differ.


Who you gonna call?


github.com/reproducible-containers

# `diffoci`

Compares two images and reports their differences.

```
$ diffoci diff --semantic --report-dir diffs/ <image1> <image2>
```

- `--semantic`: Check file contents, not metadata.
- `--report-dir`: Dump differences for manual inspection.

## `diffoci` output analysis

```
TYPE    NAME                              INPUT-0          INPUT-1
File    var/log/apt/term.log              f78e67af...      e3b6ff49...
File    var/cache/ldconfig/aux-cache      c0bd5b8e...      37f21a52...
File    var/log/dpkg.log                  fe4e8edc...      530173ca...
[...]
```

Ok, it's some APT logs. What are the differences?

# `diffoci` output analysis

```
$ diff diffs/input-{0,1}/layers-1/var/log/apt/term.log
```

```
- Log started: 2026-01-22  09:58:37
+ Log started: 2026-01-22  09:59:38
```

- `SOURCE_DATE_EPOCH` affects the filesystem metadata (mtime).
- It does **not** affect the clock of the processes running *inside* the container.
- Well, just remove the logs, right?

# The importance of being ~~idle~~ archived

```
$ docker run --rm <image> apt-cache policy gcc
[...]
 *** 4:12.2.0-3 500
        500 http://deb.debian.org/debian bookworm/main amd64 Packages
```

- http://deb.debian.org lives and breathes
- Packages and their dependencies can be updated at any time

# The importance of being ~~idle~~ archived

Solution:

- Use https://snapshot.debian.org instead of the main repo
- Pin to a specific point in time (`SOURCE_DATE_EPOCH`)
- Remove APT caches and logs after package is installed

# `repro-sources-list.sh`

- Automatically configures `/etc/apt/sources.list`
- Maps the base image timestamp to a Debian snapshot
  - Alternatively, uses `SOURCE_DATE_EPOCH`
- Ensures all future `apt-get` calls use that snapshot
- Safe to cache and reduce load on snapshot servers

# `repro-sources-list.sh`

## Non-reproducible

```
FROM debian:bookworm-20230904-slim
RUN apt-get update && apt-get install -y gcc
```

## Bit-for-bit reproducible

```
FROM debian:bookworm-20230904-slim
ENV DEBIAN_FRONTEND=noninteractive
RUN \
  --mount=type=cache,target=/var/cache/apt,sharing=locked \
  --mount=type=cache,target=/var/lib/apt,sharing=locked \
  --mount=type=bind,source=./repro-sources-list.sh\
,target=/usr/local/bin/repro-sources-list.sh \
  repro-sources-list.sh && \
  apt-get update && \
  apt-get install -y gcc && \
  rm -rf /var/log/* /var/cache/ldconfig/aux-cache
```

Digest should always be

**`sha256:b0088ba0110c2acfe757eaf41967ac09fe16e96a8775b998577f86d90b3dbe53`**

Who Reproduces the reproducible images?

WE DO!

# Litmus test for non-reproducibility

```
RUN adduser --disabled-password user
```

Unless you *really* know what you're doing, this image will stop being reproducible in a day from now.

# Litmus test for non-reproducibility

Here's why:

```
$ cat /etc/shadow
[...]
user:!:20483:0:99999:7:::
```

- `20483` is days since epoch when the user was created
- `adduser ... && chage -d 99999` is a reproducible way to scrub this

# Continuous verification

- We need some sanity checks
- So we created a CI job that rebuilds this image nightly

```
./repro-build analyze image.tar --expected-image-digest \
  sha256:b0088ba0110c2acfe757eaf41967ac09fe16e96a8775b998577f86d90b3dbe53
✅ Image digest matches
```

# Continuous verification

... doing the same thing every night

```
./repro-build analyze image.tar --expected-image-digest ...
✅ Image digest matches
```

# Continuous verification

... *every* *single* *night*

```
./repro-build analyze image.tar --expected-image-digest ...
✅ Image digest matches
```

# Continuous verification

... over and over

```
./repro-build analyze image.tar --expected-image-digest ...
✅ Image digest matches
```

# Continuous verification

... over and over and over

```
./repro-build analyze image.tar --expected-image-digest ...
✅ Image digest matches
```

# Continuous verification

but then:

```
./repro-build analyze image.tar --expected-image-digest ...
❌ Image digest does not match
```

# What happened?

```
    "rootfs": {
      "type": "layers",
      "diff_ids": ["sha256:341de903..."]
-   }
+   },
+   "variant": "v8"
```

- Buildkit v0.20.0 regression
- Small enough, but changed the image digest
- Filed moby/buildkit#5774 and it was fixed immediately

# Introducing `repro-build`

A collection of helpers for reproducible images

## The mission of `repro-build`

- Provide an easy way to build and verify images
- Avoid common mistakes, build on any environment
- Prevent reproducibility rot via continuous verification.

# Python script: `repro-build`

```
./repro-build build --source-date-epoch 0 .
```

- No dependencies
- Pins **Buildkit** to a specific version ( `v0.19.0@sha256:...` ).
- Enforces strict build parameters
- Removes common sources of non-determinism (build provenance, timestamps)
- Works with both Docker and Podman as CLIs.

# GitHub action: `freedomofpress/repro-build@v1`

```yaml
- name: Reproducibly build and push image
  uses: freedomofpress/repro-build@v1
  with:
    tags: ghcr.io/my-org/my-image:latest
    source_date_epoch: 1677619260
    push: true
```

Can replace `docker/build-push-action` for simple builds.

# GitHub action: `freedomofpress/repro-build/verify@v1`

```yaml
- name: Verify image reproducibility
  uses: freedomofpress/repro-build/verify@v1
  with:
    target_image: ghcr.io/my-org/my-image:latest
    #expected_digest: "sha256:..."
    platforms: linux/amd64
    source_date_epoch: 1677619260
    file: Dockerfile
```

Rebuild image locally, compare with remote image or fixed digest

# Reproducible containers

- Nightly Debian Bookworm and Trixie builds from snapshot archives
  - e.g., `ghcr.io/freedomofpress/repro-build/debian:trixie-20260129`
  - mostly for testing purposes

Oh, and you bet we still reproduce

**`sha256:b0088ba0110c2acfe757eaf41967ac09fe16e96a8775b998577f86d90b3dbe53`**

# There's still stuff to do

1. **Self-Describing Images**: Include the build environment and instructions *inside* the image metadata.
2. **Community Ecosystem**: A systematic way for the container community to verify and attest images (like Debian's CI).
3. **Improved Ergonomics**: Better support for multi-arch and complex build pipelines.

# Questions?

## tl;dw

- Reproducible images thwart supply chain attacks
- They will rot if no one verifies them
- We do that and so can you
- Try `freedomofpress/repro-build` in your projects

# Get involved!

Issues and PRs super welcome:

🐙 freedomofpress/repro-build

Dust off your RSS reader:

🌐 https://dangerzone.rocks/news

Ixnay on the algorithm:

🐘 social.freedom.press/@dangerzone