Write-up: Pick One

Executive Summary

During the analysis of the provided build artifact (image.tar), a hidden flag was recovered from the underlying image layers. Although the artifact was intended to have sensitive data removed before release, the use of standard Docker-style layering meant that the data remained accessible in the history of the image.

Recovered Flag: CTF{layers_never_forget}

Technical Analysis

1. Artifact Identification

The provided file image.tar was identified as a Docker image export. Upon extraction, the manifest revealed a multi-layered structure consisting of three distinct layers:

- **Layer 1:** Base Alpine Linux filesystem.
- **Layer 2:** Application/Data layer containing flag.txt.
- **Layer 3:** Final configuration layer.

2. Layer Extraction

Each layer was extracted individually to inspect the filesystem changes at each step of the build process.

Layer ID	**Key Contents**	**Observation**
Layer 1	/bin, /etc, /usr, etc.	Standard Alpine Linux base.
Layer 2	flag.txt	Contains the string CTF{layers_never_forget}.
Layer 3	.wh.flag.txt	A "whiteout" file used to hide flag.txt.

3. The "Removal" Mechanism

The analysis confirmed that the sensitive data was "removed" by deleting the file in a subsequent layer. In Docker-compatible storage drivers, deleting a file from a previous layer creates a **whiteout file** (prefixed with `.wh.`).

> ***Security Note:*** *Whiteout files do not delete the data from the image; they only instruct the container runtime to hide the file in the final merged view. The original data remains fully intact in the layer where it was first created.*

Conclusion

The flag was successfully recovered by inspecting the individual layers of the build artifact. This case highlights a common misconfiguration in containerized deployments where sensitive files are added and then deleted in separate `RUN` commands, leaving the data permanently stored in the image's history.

To properly remove sensitive data, it must be handled within the same layer (the same `RUN` command) or excluded entirely using `.dockerignore` or multi-stage builds.