🚀 Writeup - Challenge "Dockerflag" (404CTF)

📝 Introduction

Le challenge **"Dockerflag"** nous fournit une image Docker contenant un site web. L'objectif est de découvrir les secrets cachés dans cette image, en particulier un flag.

🥷 Étape 1 : Extraction de l'image Docker

- J'ai commencé par extraire l'image Docker fournie avec la commande suivante :

```
tar -xvf dockerflag.tar -C docker_image_extracted/
```

- L'image Docker contenait plusieurs fichiers `.tar.gz`, représentant les différentes couches de l'image.
- J'ai ensuite extrait chaque couche pour explorer le système de fichiers complet :

```
mkdir docker_image_layers/
tar -xzvf layer.tar.gz -C docker_image_layers/
```

🗂️ Étape 2 : Analyse de l'application

🔍 Exploration de l'application Flask

- Dans le dossier `/app/`, j'ai identifié les fichiers suivants :
  - `app.py` : Le code source de l'application Flask.
  - `requirements.txt` : Les dépendances de l'application.
  - `.git/` : Un historique Git complet, révélant les anciennes versions.
  - `index.html` : La page d'accueil du site.

🔍 Analyse du code `app.py` :

```
import os
```

```
from flask import Flask, render_template
from dotenv import load_dotenv

load_dotenv()
SECRET_KEY = os.getenv("SECRET", default="WHERE IS ZE DOTENV ?")

app = Flask(__name__)

@app.route('/')
def index():
    return render_template("index.html")

app.run(debug=False, host="0.0.0.0", port=5000)
```

✅ Analyse :

- L'application Flask charge une variable `SECRET` depuis un fichier `.env`.
- En cas d'absence de `.env`, la valeur par défaut est : **"WHERE IS ZE DOTENV ?"**.
- Cette variable est essentielle pour la sécurité de l'application.

🔍 Étape 3 : Exploration de l'historique Git

- En explorant le dossier **`.git/`**, j'ai trouvé les logs Git avec la commande :

```
git log
```

- Les logs ont révélé plusieurs commits par **Alba Laine** :
  - `commit: Source code of website`
  - `commit: Last commit before week-end !`
  - `commit: Add static ressources`
  - `commit: Requirements of website`
  - `commit: Add HTML website`

🔍 Analyse des objets Git

- J'ai exploré les objets Git (`.git/objects/`) pour récupérer les anciennes versions des fichiers.
- En décompressant les objets, j'ai découvert un fichier `.env` avec un secret :

```
SECRET="404CTF{492f3f38d6b5d3ca859514e250e25ba65935bcdd9f4f40c124b773fe536fee7d}"
```

✅ Conclusion

- Le flag a été trouvé dans une ancienne version de l'image Docker, dans l'historique Git.
- Cette méthode montre l'importance de vérifier les historiques Git lors des audits de sécurité.
- Les secrets supprimés peuvent rester accessibles dans l'historique des versions.

🚩 Flag final :

```
404CTF{492f3f38d6b5d3ca859514e250e25ba65935bcdd9f4f40c124b773fe536fee7d}
```

🚀 Commandes clés utilisées :

```
# Extraction de l'image Docker
tar -xvf dockerflag.tar -C docker_image_extracted/

# Exploration des couches
tar -xzvf layer.tar.gz -C docker_image_layers/

# Analyse de l'historique Git
git log

# Exploration des objets Git pour les secrets
zlib.decompress(open('object_path', 'rb').read())
```

📝 Remarques :

- Ne jamais laisser des fichiers sensibles dans l'historique Git.
- Toujours vérifier les anciens commits et les fichiers supprimés dans les images Docker.
- Utiliser des outils comme `.gitignore` pour protéger les fichiers sensibles lors du développement.