CTF Challenge: Espresso Grid

A photo hides more than what meets the eye. Not everything is where you first look.

Flag: ACNCTF{W4k3UpN0Cr0iss4nt}

Overview

1. Check the file for appended data (archive signature).]
3. Extract hint.txt from 7z compressed file.
4. Follow the hint → OSINT to find the dish name.
5. Extract the LSB-hidden menu.txt.enc from image pixels.
6. Decrypt menu.txt.enc with the dish password.
7. Convert hex to ASCII to get the flag.

Solver's Path

Quick checks (useful first commands)

Linux / Kali

```
file 1.png
ls -lh 1.png
binwalk 1.png          # quick scan for appended archives
strings -n 8 1.png | head -n 50
```

Windows (PowerShell)

```
Get-Item 1.png | Format-List Name,Length
# You can use a Windows binwalk build or inspect bytes with a hex viewer.
```

Auto extract with binwalk (Linux)

```
binwalk -e 1.png
```

Look in _image.png.extracted/ for files ([filename].7z , [filename].zlib ,etc.)

Extract hint.txt from hint.7z (Password Protected)

Linux: 7z x [filename].7z or .zlib

Windows: Right-click → 7-Zip → Extract Here

Read hint.txt — it points to Instagram / OSINT clue

```
cat hint.txt (Linux)
type hint.txt (Windows)
```

Extract LSB-hidden file from pixels

```
from stegano import lsb
hidden = lsb.reveal('1.png')
with open('menu_extracted.enc','wb') as f:
  f.write(hidden.encode('latin1'))
print('wrote menu_extracted.enc')
```

Decrypt menu_extracted.enc

```
openssl enc -d -aes-256-cbc -in menu_extracted.enc -out decrypted.txt -pass
pass:"croissant"
```

Simply use online decryption tools to get the flag in flag.txt.

Quick troubleshooting notes

- **Finding the appended archive (hint.7z):**
  - Some solvers may only try `steghide` and miss the appended 7z archive.
  - They'll need to notice the PNG is oversized and use `binwalk` or manual carving.
  - If binwalk doesn't show proper results then just open the image with 7z file manager
- **Correct password for extraction:**
  - Password "muse" (for the 7z stage) is OSINT-derived, not guessable.
  - If they don't dig into the café clue, they'll stall.
- **Revealing the LSB payload:**
  - Solvers may expect `steghide` again, but the second payload is embedded with pixel-level LSB steganography.
  - They'll need to pivot to tools like `stegano` or write extraction code.
- **Decryption mismatches:**
  - Password "croissant" (for the LSB stage) is also OSINT-derived (based on hint.txt), not guessable.
  - Using `openssl` requires the *exact* options used for encryption (`-aes-256-cbc`, same password, no `-pbkdf2`).
  - A small mismatch gives "bad decrypt" errors.