CTF WRITEUP · CRYPTO / FORENSICS / MISC

# ByteMe CTF 2026

**Team:** 0xAvengers · **Event:** ByteMe CTF 2026 (OWASP PCCOE) · **Result:** 5 solved, 4 attempted

---

## Soul Forge (crypto, two-time pad)

**Category:** Cryptography

Two "Soul Forge transmissions" encrypted under the same reused keystream, the classic two-time pad: XOR the two ciphertexts together and the keystream cancels out, leaving the XOR of the two plaintexts. With enough structure in one plaintext (known headers, repeated formatting, or a crib), that's enough to peel both messages apart byte by byte and recover the hidden fragment.

---

## Soul Vault metadata recovery (forensics)

**Category:** Forensics

`vault.log` documented a checksum error across two recovery blocks, with no checksums actually given to compare. The real payload was simpler than the log implied: one block of hex sitting between two marker strings (`00ff11aa` ... `deadbeef`) in the accompanying file. Decoding that hex directly, no block comparison needed, gives the flag.

**Flag:** `OWASP{br0k3n_s0ul_v4ult_r3c0v3r3d}`

---

## The Soul Stone Remembers (crypto, ECDSA nonce reuse)

**Category:** Cryptography

A vault signs messages with ECDSA, and two signatures shared the same random nonce `k`, the single most common real-world ECDSA failure. Given two signatures `(r, s1)` and `(r, s2)` over known messages with the same `r` (meaning the same `k`), the nonce and then the private key `d` solve out directly from the signature equations.

Recovering `d` wasn't the end of it: the vault's actual encryption key was `sha256(b"SOUL-VAULT:" + str(d).encode())`, a salted derivation rather than `d` used raw, and the cipher underneath was a plain repeating-key XOR, not an authenticated scheme, which is why naive AEAD-decryption attempts

against it never worked. Once both of those were identified, decrypting the vault was immediate.

---

## Three Soul Stone fragments (forensics, LSB steganography)

**Category:** Forensics

Three separate images (`soul_fragment.png`, `soul_world.png`, `last_soul.png`), each hiding a message in the least significant bit of each RGB channel, read row-major, most significant bit first. The first two decode directly to the flag text. The third decodes to a base64 string first, one extra layer, which then decodes to the final flag.

**Flags:** `OWASP{h1dd3n_s0ul_b3n34th_th3_sh4d0w}`, `OWASP{s0ul_w0rld_h1d3s_1ts_s3cr3t}`, `OWASP{th3_l4st_s0ul_w4s_n3v3r_l0st}`

---

## Attempted, not solved

- **Soul Vault rate limiter** (misc): a "collect fragments" endpoint that returns 403 after repeated hits, and the 403 never lifted across more than an hour of polling, on both my own IP and (apparently) everyone else's. Tried every forwarded-IP header variant, path tricks, and protocol variants; none of them changed the response, which points at the limiter being keyed on something upstream (likely the proxy's own address) rather than the client. Left a background poller running rather than call it solved.
- **The impostor trail** (OSINT): a challenge built around finding a CTF's genuine first social-media announcement versus an impersonating account. Inconclusive; the one alias lead I had didn't resolve to anything real.
- **House of Water** (pwn): a heap challenge with a secret loaded into an unreachable chunk at start. Didn't find the bug in time.
- **A personalized OSINT flag** (`OWASP{FIRSTNAME_LASTNAME_PRN_ROLE}`): required finding a specific person's PRN number from public social profiles. Left open.