

# ROUNDING INFLATION ATTACK

## Root Cause

- The vault uses:

```
shares = (amount * total.shares) / total.amount; // rounds down
```

- **Shares minted are truncated** (floor division), but `total.amount` is always increased by the full deposit.
  - If `total.shares` is very small and `total.amount` much larger than `amount`, the computed `shares` can round down to zero.
  - Each such deposit **increases** `total.amount` **but not** `total.shares`, inflating the *share price*.
- 

## “classic inflation” defense didn’t help

- The vault does **not** use `ERC20(asset).balanceOf(address(this))` to measure assets.
  - It tracks `total.amount` internally, only updated via deposit/withdraw functions.
  - So, simply transferring tokens in (classic donation) won’t work.
  - However, the **round-down minting** *itself* becomes a donation — a *stealth donation* — when the minted shares are zero.
- 

## How the attack snowballs

The attacker:

1. Starts with `total.shares = 1` and `total.amount = 2`.
2. Deposits `1` → gets `0` shares → `total.amount = 3`.
3. Deposits `2` → gets `0` shares → `total.amount = 5`.
4. Deposits `4` → gets `0` shares → `total.amount = 9`.
5. ...

This follows:

```
amount_n = 2^(n-1)
total.amount grows exponentially
```

```
total.shares stays fixed
share price skyrockets
```

With enough iterations, a **victim's deposit yields them 0 shares**, meaning their deposit is “donated” to existing shareholders — in this case, the attacker holding the single share.

---

## General Pattern

This is a **share-price manipulation via repeated zero-share deposits**, caused by:

- Integer truncation bias during share minting.
  - No **minimum share mint guarantee**.
  - Very low initial share supply.
- 

## Security Lessons (applies to ERC-4626 & vault math)

### 1. Always mint at least 1 share for any nonzero deposit

- Use:

```
if (shares == 0 && amount > 0) shares = 1;
```

or revert instead of silently donating.

### 2. Seed vaults with a nontrivial initial liquidity

- A small deposit from the deployer sets a healthy price ratio.
- Prevents attacker from creating `total.shares = 1` state.

### 3. Handle rounding direction consciously

- Rounding down is usually safer for free-mint prevention, **but** you must address the “free donation” side effect.
- Some protocols use **round to nearest** with min-share guarantees.

### 4. Test empty or near-empty vault scenarios

- Many inflation / donation exploits only appear when vault is empty or has extremely skewed price ratios.

## 5. **Be wary of isolated-pair deployments**

- Every new pool starts in a vulnerable “bootstrapping” state unless explicitly seeded.