Unstake() transfers `amount + reward` instead of just reward

```
/// @dev Allow user to unstake their RITE after staking duration
    /// @param _index The index of the stake
    function unstake(uint _index) external {
        require(stakes[msg.sender].length > 0, "Staking: no stake");
        require(_index < stakes[msg.sender].length, "Staking: invalid stake");
        require(
            block.timestamp >= stakes[msg.sender][_index].endAt,
            "Staking: staking is not ended yet"
        );

        uint256 amount = stakes[msg.sender][_index].amount;
        uint256 reward = (amount * APY) / 100;
        require(
            ERC20(RITE).balanceOf(self) >= amount + reward,
            "Staking: insufficient balance"
        );

        ERC20(RITE).safeTransfer(msg.sender, amount + reward);
        emit Unstaked(
            msg.sender,
            amount,
            block.timestamp,
            stakes[msg.sender][_index].month
        );

        stakes[msg.sender][_index] = stakes[msg.sender][
            stakes[msg.sender].length - 1
        ];
        stakes[msg.sender].pop();
    }
```

Code in question:

```
uint256 amount = stakes[msg.sender][_index].amount;
uint256 reward = (amount * APY) / 100;
require(
    ERC20(RITE).balanceOf(self) >= amount + reward,
    "Staking: insufficient balance"
);

ERC20(RITE).safeTransfer(msg.sender, amount + reward);
```

Intended logic

- User **stakes tokens** → tokens are transferred from user → contract.
- On **unstake**, user should receive:
  - Their original **staked tokens** back.
  - **Reward** (based on APY).

So expected payout = `amount (already staked)` + `reward`.

Actual logic

the staking function never transferred `amount` to the contract. That means:

- Contract **never received the staked tokens**.
- But on `unstake`, it **sends** `amount + reward`, effectively minting `amount` out of nowhere.

🎯 Impact

- **Users can drain the staking contract**:
  - Stake 100 tokens (but not actually transferred).
  - On unstake, contract pays `100 + reward`.
  - Net profit = 100 tokens + reward for free.
- This breaks the whole economics of staking since **no one is actually locking tokens** but everyone can withdraw tokens from the contract treasury.

Exploit Scenario

1. User calls `stake(100)`.
   - No `ERC20.transferFrom()` → contract balance unchanged.
2. Wait until lock ends.
3. Call `unstake()`.
   - Contract transfers `100 + reward`.
   - Free 100 tokens minted.
4. Repeat until contract is drained.

Fix

There are **two options** depending on design:

Option 1: Proper staking

In `stake()`, actually **transfer tokens into contract**:

```
ERC20(RITE).safeTransferFrom(msg.sender, address(this), amount);
```

Then `unstake()` logic (returning `amount + reward`) is correct.

Option 2: Reward-only staking (if intended)

If the staking model is “virtual staking” where users don’t transfer tokens in, then **only reward should be paid**:

```
ERC20(RITE).safeTransfer(msg.sender, reward);
```