Unsafe ERC20 Transfers via `transfer` - `transferFrom` Instead of `safeTransfer`

Explainer

In Ethereum smart contracts, ERC20 token transfers are typically done using `transfer()` and `transferFrom()`. However, **not all ERC20 tokens strictly follow the standard** — some do not return a boolean, some return nothing, and others may even revert silently on failure. Using `transfer()` directly in such cases can result in **undetected failed transfers**, loss of funds, or contract logic proceeding incorrectly.

To mitigate this, libraries like OpenZeppelin provide **`safeTransfer()` and `safeTransferFrom()` wrappers**, which **enforce success checks** and **support non-standard tokens**.

Cause

1. Developers call:

   ```
   token.transfer(to, amount);
   ```

   or

   ```
   token.transferFrom(from, to, amount);
   ```

   assuming that:
   - These functions always return `true` on success.
   - The token reverts or errors out clearly on failure.
2. But some tokens (like USDT) return nothing, and others may revert silently, causing:
   - Funds to not be transferred,
   - Code to continue execution without detecting the failure,
   - Resulting in **incorrect accounting, loss of funds, or security vulnerabilities**.

Where to Look

1. **Any contract interacting with ERC20 tokens**, especially:

    - Vaults
    - Bridges
    - Token wrappers
    - Lending/Borrowing systems
    - Reward distribution contracts
2. **Raw calls to `transfer()` or `transferFrom()` without return value checks**, like:

    ```
    token.transfer(user, amount); // 🚨 unsafe
    ```

3. Contracts **interacting with older or widely-used non-compliant tokens**, like:
    - USDT (Tether)
    - BNB
    - Other bridged tokens or L2-specific wrappers

Why This Happens

- The ERC20 spec is loosely defined in the original standard (EIP-20).
- Real-world implementations often diverge.
- Developers assume all tokens comply strictly with the spec, and skip return checks.
- Testing often uses well-behaved mocks that don’t expose this risk.

Recommended Solutions

1. **Always use `safeTransfer()` and `safeTransferFrom()` from OpenZeppelin's SafeERC20**

    ```
    import { SafeERC20 } from
    "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

    using SafeERC20 for IERC20;

    token.safeTransfer(to, amount);
    token.safeTransferFrom(from, to, amount);
    ```

2. **Avoid trusting raw `transfer` / `transferFrom` return values**
    - If you must use them, manually check:

        ```
        require(token.transfer(to, amount), "Transfer failed");
        ```

3. **Test with non-standard tokens** like USDT during development.
4. **Auditors** should flag all `token.transfer(...)` and `token.transferFrom(...)` usages that:
    - Do not check return values, or
    - Don’t use a SafeERC20 wrapper.
5. **Protocol teams** should enforce this in linting rules, static analysis, or code reviews.