<u>**Popsicle Finance Bug**</u>

Popsicle finance is an investment platform that acts and liquidity provider at several liquidity pools and manage to optimize gained fees for the benefit of the investor. In August 2021, a vulnerability of the Popsicle Finance smart contract *SoberttoFragola* was exploited, and led to a 25M$ loss of users funds.

In short, the bug is that the popsicle finance team forgot to override their default ERC20 transfer function, which decrease x tokens for user u_1 and increase x tokens to user u_2. But differently from other tokens that implements the ERC20, the Popsicle's token was designed such that every user had a different amount of fees to collect (which is per token), based on its entry date and the total fees that are already collected.

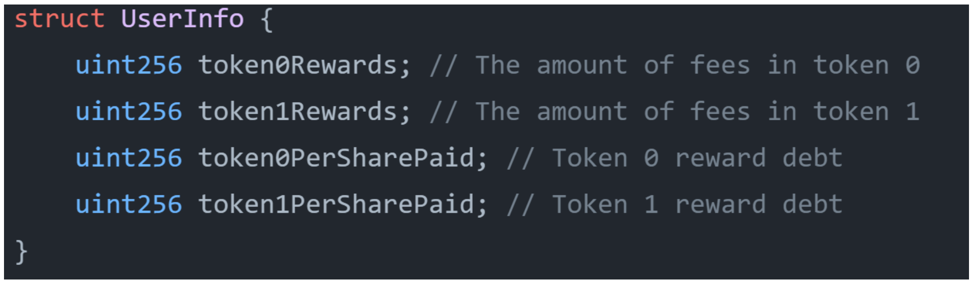
The bug let the attacker to buy M Popsicle Finance tokens and transfer the tokens to a friend that has been joined a year ago and can collect 10% fees per share. Because of the bug, the fried can collect the 10% fees reward also for the extra M tokens that he now holds. Which is a bug since this user doesn't deserve 10% fees per share for its new M transferred tokens.

The way popsicle finance records the fees the user can collect is by two fields:

- **Popsicle debt field** (in the code "`token0Rewards`" / "`token1Rewards`") – this field is not "per token" but is how much popsicle finance owns this user for different reasons. One reason might be a withdraw without claiming the earned fees.
- **Fees earned per share paid** (in the code "`token0PerSharePaid`" / "`token1PerSharePaid`") – Popsicle finance would rather prefer to maintain the field **Fees earned per share**, which is how many fees are earned for this user per share from the day it entered. But since maintaining this field requires updates for all users at every earn – it costs a lot of gas. Therefore, we add the "**paid"** and actually

$$\text{Fees earned per share} = \text{Current fees per share earned} - \text{Fees earned per share paid}$$

That means we only maintain the fees earned at the day we enter and at the time we collect the fees we get the difference per share and update this field to the current fees earned.

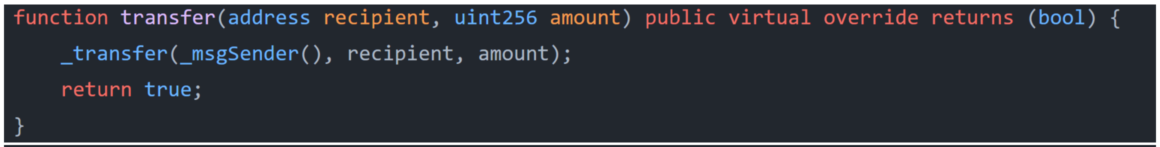


```
struct UserInfo {
    uint256 token0Rewards; // The amount of fees in token 0
    uint256 token1Rewards; // The amount of fees in token 1
    uint256 token0PerSharePaid; // Token 0 reward debt
    uint256 token1PerSharePaid; // Token 1 reward debt
}
```

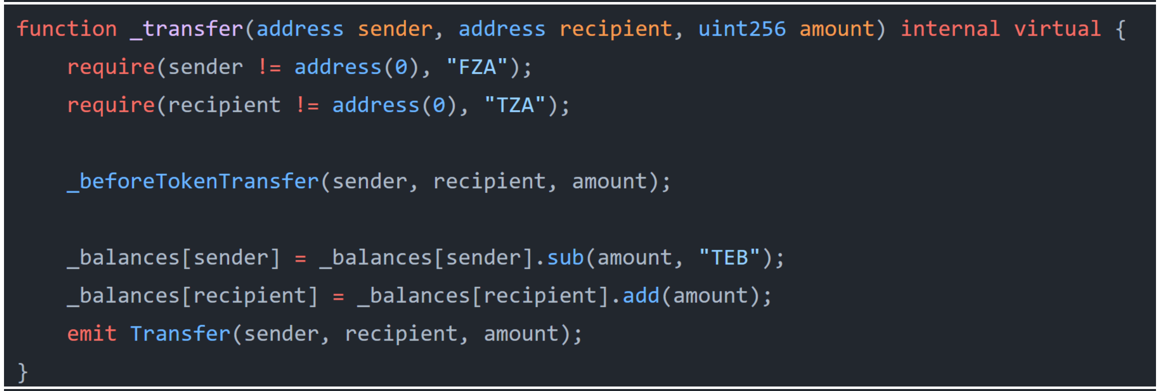
In the actual code the fees are recieved in two different tokens.

At deposit the popsicle finance code put the total earned by the user at the dept field and update **Fees earned per share paid** to the current earned fees per share which is correct for the new deposited shares.

But at transfer the popsicle finance team forgot doing that, and just transferred the tokens and let the user to claim the fees per share for much larger amount of tokens.



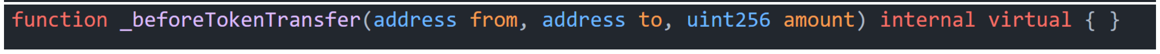
```
function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}
```



```
function _transfer(address sender, address recipient, uint256 amount) internal virtual {
    require(sender != address(0), "FZA");
    require(recipient != address(0), "TZA");

    _beforeTokenTransfer(sender, recipient, amount);

    _balances[sender] = _balances[sender].sub(amount, "TEB");
    _balances[recipient] = _balances[recipient].add(amount);
    emit Transfer(sender, recipient, amount);
}
```



```
function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
```

The transfer function doesn't update the user fees per share information.

To fix the bug, the transfer method should be implemented with the same logic as in deposit, and update the transferee information, including the **Fees earned per share paid**, into valid values.

Another bug that the attacker exploited, allows him not to have this friend who invested a year ago, but send the tokens to an address that not exists – and therefore its **Fees earned per share paid** is at the initial value which is 0. So, the attacker claims fees per share that has been earned from the beginning of the popsicle finance platform.

A rule that would catch this issue, is that calling the *transfer* method can't increase the total worth of all members. What is a member worth? It's balance + fees owned.

Clearly after transfer to a user with lower **Fees earned per share paid** value, this amount is increased, and the Certora Prover finds the instance that allows that to happen.