CONFIDENTIAL

Security report for Ice Cream

TABLE OF CONTENTS

REPORT

Farm

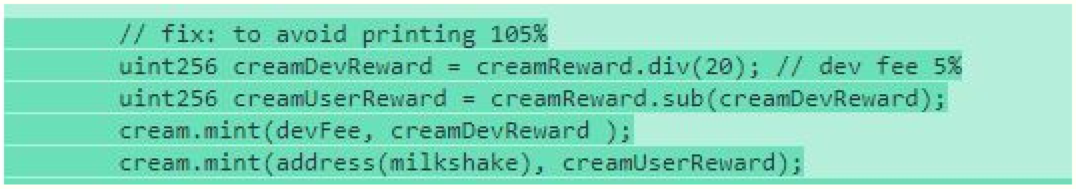
https://github.com/IceCreamSwap/contracts/tree/main/farm-contracts

CreamToken.sol

- Clone of https://github.com/pancakeswap/pancake-farm/blob/master/contracts/CakeToken.sol. No functional changes.

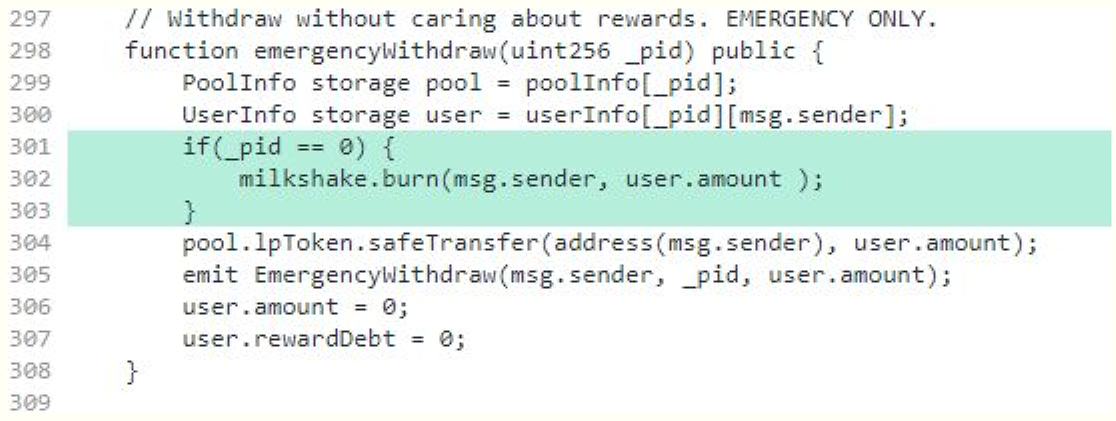
MasterChef.sol

- Owner of the contract has ability to do the following:
    - add new pools
    - modify existing pools' `_allocPoint`
    - `updateMultiplier`
    - `updateBonus`
    - `updateCreamPerBlock`
    - `setDevFee`
    - `setTaxAddr`
    - `setTax`
- The access control for the functions to be called by the owner are enforced by the onlyOwner modifier. The checks in the previous version which use require and have more than 1 owner (governance and owner) address have all been removed. There is only 1 owner now.
- Additional checks in `updateMultiplier` and `updateCreamPerBlock` are added. Multiplier cannot be set to be above 3, and `creamPerBlock` cannot be set to be above 4 ETH. This will limit setting the emission per block to an absurdly high number, but a pool with an absurdly high `allocPoint` will still be able to get most of the entire farm's emission per block..
- Migrator function has been removed.
- Fix added to mint 100% of the block reward instead of 100% + 5% (dev fee is included as 5% of the 100%) like in other clones of MasterChef



```
        // fix: to avoid printing 105%
        uint256 creamDevReward = creamReward.div(20); // dev fee 5%
        uint256 creamUserReward = creamReward.sub(creamDevReward);
        cream.mint(devFee, creamDevReward );
        cream.mint(address(milkshake), creamUserReward);
```

- The syrup (milkshake) bug has been fixed in emergencyWithdraw by forcing a burn of Shake tokens equal to the same amount of ICS withdrawn by the user. If the user does not have enough Shake tokens, the function will revert.



```
    // Withdraw without caring about rewards. EMERGENCY ONLY.
    function emergencyWithdraw(uint256 _pid) public {
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        if(_pid == 0) {
            milkshake.burn(msg.sender, user.amount );
        }
        pool.lpToken.safeTransfer(address(msg.sender), user.amount);
        emit EmergencyWithdraw(msg.sender, _pid, user.amount);
        user.amount = 0;
        user.rewardDebt = 0;
    }
```

- When the MasterChef contract is deployed, the owner has to be verified to be the timelock contract, and not an EOA.

MilkShake.sol

- Clone of https://github.com/pancakeswap/pancake-farm/blob/master/contracts/SyrupBar.sol, but with some functional changes made.
- New functions include taxation of harvests
- `setTax`, `setTaxAddr` are both functions used to set and change the tax amount and destination address. Can only be called by owner (masterchef contract), which is done with `onlyOwner` modifier check. Maximum value for tax can be set to 100 (10%)
- taxUser is a private function used in `safeCreamTransfer` to deduct an amount of harvested farm tokens which will be set to the `taxAddr`. This only affects the harvested cream, not the deposited LP token.
- The event `MilkShakeTransfer` is defined twice with different arguments. While few things may need this event, it's still recommended to have a single compatible definition.

SmartChef.sol

- SmartChef has only 1 pool, instead of an array of pools. This means that each SmartChef deployed contract can only support 1 token to stake, and 1 reward token.
- Owner can call the following functions, with access control done by onlyOwner modifier:
    - `startReward`
    - `stopReward`
    - `emergencyRewardWithdraw`
- An explanation on why the SmartChef needs to have functions to arbitrarily stop or restart rewards would be helpful.
- A `_governance` address variable was added, but does not appear to be used.

CreamRecovery.sol

- After deployment, ownership has to be renounced to 0x0 to prevent additional minting.

Timelock.sol

- 24 hours time lock. Min delay is 6 hours, so it can be lowered to that, but the `setDelay` is also timelocked, and requires it to be called from the timelock contract itself.
- Change made to only allow 1 admin instead of multiple admins previously, for time lock.

Swap

https://github.com/IceCreamSwap/contracts/tree/main/swap-contracts
(Compared with https://github.com/sushiswap/sushiswap/blob/master/contracts/uniswapv2/)

UniswapV2ERC20.sol

- Same functionality as Sushiswap.

UniswapV2Factory.sol

- Same functionality as Sushiswap, migrator related code removed

UniswapV2Pair.sol

- Similar to Sushiswap
- uint denominator = rootK.mul(15).add(rootKLast); in _mintFee. Using 15 instead of 5, which is used in sushiswap, which means out of the 0.30% fees, 0.15% is for liquidity providers, and 0.15% is for the dev fund. This is clarified here: https://ice-cream-swap.gitbook.io/icecreamswap/roadmap/icecreamswap-exchange
- Migrator code portions removed

UniswapV2Router02.sol

- Same as Sushiswap

ISSUES

ICS-001: devFee address variable is misnamed as devFee

Severity: Info

The `devFee` variable is actually the address that the devFee is sent to, instead of the amount of `devFee` percentage.

Recommendations

It is recommended to rename `devFee` to `devFeeAddr` to reflect this.

ICS-002: Unused governance address in SmartChef.sol

Severity: Info

A `_governance` address variable was added to SmartChef.sol, but does not appear to be used.

Recommendations

Remove the governance address variable from the smart contract.

ICS-003: Remove unnecessary commented out code

Severity: Info

In UniswapV2Pair.sol, there is a `fee_to` variable that is commented out.

Recommendations

Remove unused comments such as `fee_to` for code readability purposes.

ICS-004: SmartChef owner can arbitrarily stop or restart rewards

Severity: Info

In SmartChef.sol, there are 2 functions; `startReward` and `endReward`. Both of which can be called by the owner to set the value of `startBlock` and `endBonusBlock`.

Recommendations

If there is no need for such functionality to stop/restart rewards, the above mentioned functions should be removed. Otherwise, explain in the documentation why such functionality is required.

ICS-005: MilkshakeTransfer event is defined twice

Severity: Info

The `MilkshakeTransfer` event is declared twice, with different function arguments.

```
event MilkShakeTransfer(address indexed user, uint256 amount, uint256 tax);

event MilkShakeTransfer(address _to, uint256 _total, uint256 _amount, uint256
_tax, uint256 creamBal);
```

In the code, the latter is used.

Recommendations

Remove the first instance duplicate event.

RISK FACTORS

- 5% of minted ICS tokens goes to the dev fund.
- 5% of ICS harvests are taxed and goes to a tax address. This could be raised up to 10% of ICS harvests.
- In the documentation (https://ice-cream-swap.gitbook.io/icecreamswap/roadmap/tokens-distributions), it is mentioned that taxed funds are burnt. The taxed funds could be used for other purposes other than burning if the destination address is an account that can arbitrarily do token transfers.
- There is a lack of any test coverage for any of the smart contracts provided.

RECOMMENDATIONS

- It is recommended to send the tokens to a burn address. This will ensure that the funds will definitely be burnt.
- As this is a fork of Pancake Swap's code, which already have some test cases (https://github.com/pancakeswap/pancake-farm/tree/master/test) , it is recommended to build on top of the existing tests and add test coverage, especially for custom code changes (e.g. tax feature).